UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Departamento de Ingeniería Informática

TRABAJO FIN DE GRADO INFORMÁTICA

DESARROLLO DE GESTOR DE COMIDAS A DOMICILIO MEDIANTE LA APLICACIÓN DE MENSAJERIA INSTANTANEA TELEGRAM

Universidad
Carlos III de Madrid

Autor- Antonio Pablo Rivero Ortiz

Tutor- Álvaro Luis Bustamante

Contenido

Índice de figuras

Índice de Tablas

1. Introducción:

Existe una tendencia cada vez mayor por parte de los consumidores en concentrar el máximo de aspectos de su vida en el terminal móvil que llevan con ellos. En respuesta a esta tendencia, las empresas están lanzando al mercado aplicaciones para poder acceder a sus servicios desde cualquier lugar mediante un Smartphone.

- Gran parte del uso que le dan los consumidores a estos terminales se centra en las aplicaciones de mensajería instantánea.
- En el pasado reciente, los servicios web relacionados con los pedidos de comida a domicilio han obtenido un gran éxito.

El sistema que se presenta pretende aprovechar estos factores y plantear al cliente una nueva forma de relacionarse con la empresa a la que va a realizar su pedido mediante el uso de la aplicación de mensajería instantánea *Telegram*.

Este sistema ofrece ventajas a los clientes, ya que pueden realizar su pedido rápidamente introduciendo tan solo unas pocas palabras, o bien tener tiempo para discutir los platos con sus acompañantes sin tener que hacer esperar al empleado encargado de atender al teléfono en el restaurante.

Para los gerentes del establecimiento, le ofrece mejoras competitivas, al contar con un sistema más eficiente para recoger los pedidos, con un coste menor a un nuevo empleado, además de una gestión más directa que con las aplicaciones alternativas.

1.1. Contexto:

Dentro del mercado en el que el sistema opera no existe todavía ningún producto con las mismas características que nosotros ofrecemos, sin embargo, si existen productos que pueden suplir las necesidades a las que está enfocado. Estas alternativas consisten en aplicaciones para móvil o páginas web en donde se encuentran métodos de contacto con los restaurantes de determinada zona, así como los menús y platos que estos ofertan. Algunos ejemplos de estas alternativas son ***LNR*** o ***JE***.

La Nevera Roja (Usaremos LNR para referirnos a ellos) o Just Eat, son empresas que ya están establecidas (Just Eat se fundó en 2001 y La Nevera Roja en 2011), el público las conoce y tienen cierta potencia empresarial (incremento de un 600% en las ventas para La nevera Roja entre 2012-2013).

Hay que tener en cuenta que los dispositivos necesarios para utilizar el sistema se encuentran ya al alcance de los usuarios, a pesar de la disminución del crecimiento de las ventas que está sufriendo este mercado.

El 80% de la población adulta en España tiene un Smartphone, lo que facilita la entrada en el negocio de las aplicaciones o el uso de estas para móviles o tablets.

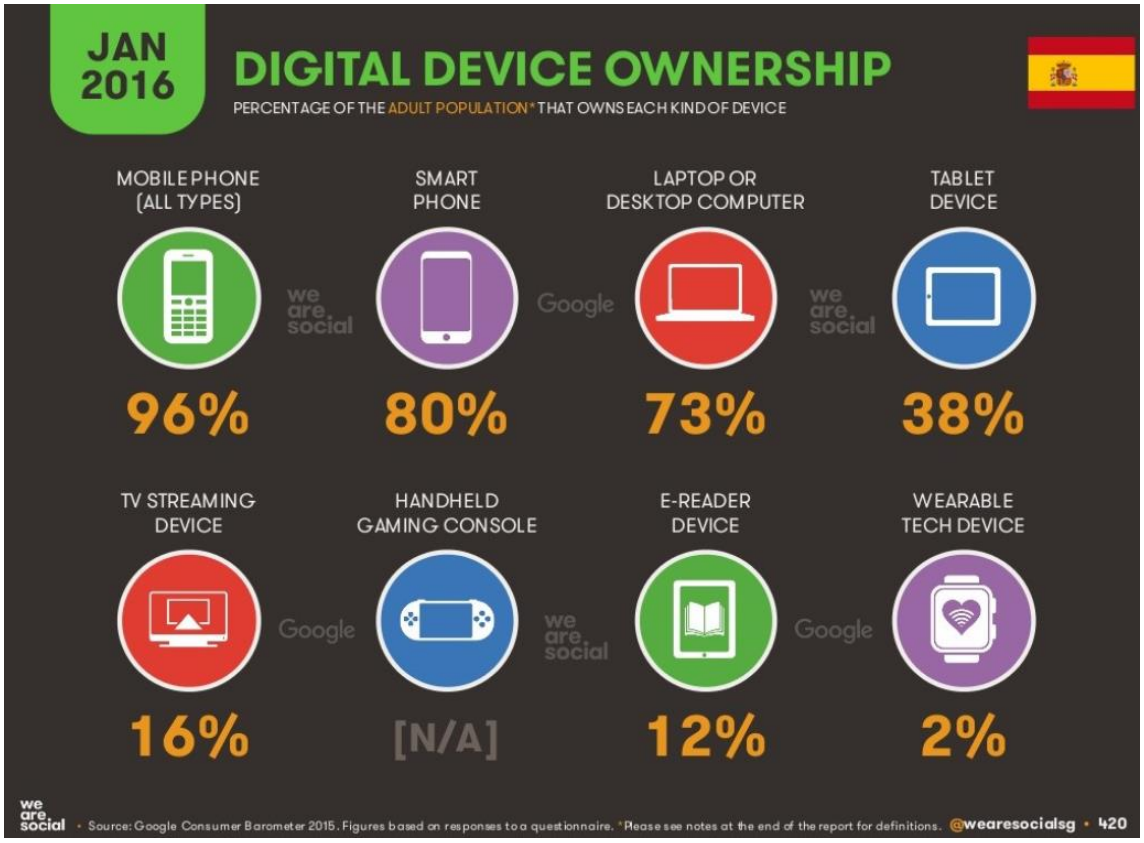


Figura 1: Porcentaje de población adulta con dispositivos tecnológicos

El sistema instalado en el restaurante puede funcionar en equipos de bajo coste, por lo que esta parte del sistema tampoco se encontrará con problemas derivados de avances o problemas tecnológicos. Este sistema está orientado a funcionar en equipos de 40 euros.

Los avances informáticos no presentan una amenaza, en todo caso tan solo ventajas, al poder ampliar las funcionalidades del sistema si los equipos disponibles acaban teniendo más potencia que los usados como referencia para realizar las pruebas.

El público objetivo del sistema son los establecimientos de comida rápida, tengan o no servicio de envío a domicilio.

Ya que atraer a los clientes de estos establecimientos forma parte esencial para nuestro negocio, es importante describir como serán los individuos para los que el equipo ha sido elaborado.

Este grupo está formado por gente joven, que esté familiarizado con el uso de aplicaciones en *Smartphones*.

Considerando que la familiaridad en el uso de aplicaciones de mensajería instantánea es la principal barrera con la que nos encontraremos para conseguir que los consumidores de comida rápida utilicen nuestro sistema, prestaremos especial atención en que toda la publicidad se oriente a hacer ver que el método que ofrecemos es tan cómodo y sencillo como comunicarse con sus conocidos a través de *WhatsApp* o cualquier otra aplicación de mensajería instantánea.

Aunque dentro del mercado de la comida rápida puede haber consumidores en un rango de edad muy amplio, desde grupos de amigos jóvenes a familias, el punto de entrada que utilizaremos es aquel miembro del grupo que se encuentra cómodo al realizar gestiones a través del móvil y no le resulta extraño que la conversación se produzca a través de una pantalla, sin hablar con nadie directamente.

Dado el planteamiento anterior, para definir los posibles grupos a los que tendremos que dar servicio, los dividiremos en dos grupos:

- El primer grupo consistirá en núcleos familiares, en donde al menos uno de los miembros servirá de punto de entrada, haciéndose cargo de gestionar el pedido para el resto de los miembros del grupo. Será vital que la accesibilidad del sistema sea tal que cualquiera de los otros miembros se sienta atraído a utilizarlo la próxima vez que vaya a realizar un pedido, es decir, al haber convencido al miembro más cercano a las nuevas tecnologías, conseguiremos acceder a más clientes por el "*boca a boca"*, sumando efectividad al proceso al realizarse entre familiares.

- El segundo grupo consistirá en grupos en los que la relación entre los miembros sea otra distinta a la familiar. Dentro de este grupo podremos encontrar desde amigos a compañeros de trabajo. De forma general, estos grupos serán más homogéneos en cuanto a familiaridad con nuevas tecnología o edad.

En cuanto al contexto legal, el manejo de los datos personales de los clientes que realicen el pedido deberá ser tratado en base a la legislación vigente actual, que en este caso está recogida en la Ley Orgánica 15/1999 del 13 de diciembre.

1.2. Objetivos:

El principal objetivo del proyecto es la elaboración de un *Script* para *Telegram*, implantado dentro de un sistema formado por una base de datos y un sistema operativo. Todo estará alojado en una Raspberry Pi.

Este sistema podrá ser instalado en establecimientos reales para obtener beneficios.

Al no tener experiencia previa en este tipo de proyectos, adicionalmente el aprendizaje del uso de las herramientas necesarias formará parte de los objetivos del proyecto, sin olvidar las *buenas prácticas* de programación y desarrollo. Para ello se consultará con la comunidad de desarrolladores de LUA, así como la documentación disponible de este lenguaje de programación.

Se pretende utilizar lo aprendido durante estos años para ser capaz de desarrollar una aplicación que sea capaz de interactuar con un sistema operativo, un programa externo y una base de datos.

1.3. Objetivos específicos:

- Elaborar el **estado del arte** para dar explicación al diseño elegido.
- Describir los **requisitos** que ha de cumplir el diseño del sistema.
- Desarrollar la **planificación** y el **presupuesto** del proyecto.
- Describir las **posibles evoluciones** futuras del proyecto.
- Explicar el **diseño** desde la óptica de la interfaz de usuario y la arquitectura y seguridad del sistema.
- Enmarcar el proyecto en el marco **socio-económico**, **legal** y **tecnológico** actual.

1.3.1. Medios:

Ha sido necesario un ordenador y un teléfono móvil Android con la aplicación *Telegram* instalada.

Para el correcto funcionamiento de la aplicación, el sistema operativo del teléfono móvil es indiferente.

1.3.2. Ordenador:

Sistema Operativo	• Ubuntu Linux 16.04
CPU	• Intel Core i7-6700HQ
RAM	• 8.00 GB Dual-Channel
Gráficos	• Nvidia GeForce GTX 960M 2GBs • Intel HD Graphics 530
Almacenamiento	• 1TB HDD

Tabla 1: Especificaciones ordenador utilizado

1.3.3. Dispositivo móvil:

Se han utilizado varios móviles para comprobar el funcionamiento del sistema cuando varios usuarios interactúan con él.

Cualquier móvil que pueda utilizar la aplicación *Telegram* puede valer para realizar esta interacción.

Según el SO del dispositivo podrán utilizarse determinadas versiones.

- **Android**: Versión 2.2 y superiores.
- **iOS**: 6 y posteriores.
- **Windows Phone**: No se especifican versiones con las que la aplicación no funcione correctamente.

2. Estado del arte:

Este apartado estudia el contexto del sistema.

2.1. Producto:

2.1.1. Mensajería Instantánea:

Es el sistema de comunicación elegido por los usuarios del sistema, por lo que es una parte esencial del proyecto.

2.1.1.1. WhatsApp:

2.1.1.1.1. Funcionamiento:

En 2009 aparece en el mercado *WhatsApp*. Esta aplicación permite a los usuarios comunicarse entre ellos y enviar archivos multimedia mediante la aplicación en sus móviles. Obra de Jan Koum y Brian Acton, dos ex-empleados de Yahoo.

En febrero de 2016 alcanzó los 1000 millones de suscriptores.

A pesar de su éxito, en un comienzo presenta serias vulnerabilidades de seguridad al enviar los mensajes como texto plano, lo que permitía cambiar el estado de algunos usuarios desde determinadas páginas web, además de hacerle vulnerable a un ataque *Man in the middle*.

En abril de 2016 se incluye cifrado de extremo a extremo para solucionar estos problemas.

El sistema de cifrado que se utiliza guarda las claves privadas en los propios terminales, esto impide que cualquier agente malicioso pueda acceder a las claves de los usuarios consiguiendo acceso a los servidores de *WhatsApp*. Adicionalmente a esta medida, *WhatsApp* utiliza *TextSecure*, una aplicación de encriptación de mensajes desarrollada por **Open Whisper Systems**.

Esta aplicación utiliza tres tipos de claves públicas (una para identificar al dispositivo, otra generada periódicamente y firmada digitalmente por la anterior, y otra que se usa solo una vez en cada utilización del servicio) y tres tipos de claves de sesión (una clave de administrador usada para generar la clave de cadena, una clave de cadena utilizada para crear la clave de mensaje, y una clave de mensaje que consta de 80 bytes: 32 de ellos para una clave AES-256, otros 32 para una clave HMAC-SHA256, y otros 16 para un IV, un vector de inicialización).

2.1.1.1.2. Carencias:

La carencia principal que tiene la aplicación es la imposibilidad de utilizar una API o algún método sencillo para poder integrarla en otros sistemas o conseguir que se comunique con otros procesos en un equipo. Aunque es posible utilizar una versión de escritorio de la aplicación o acceder a la versión de Android mediante simuladores, el esfuerzo necesario para obtener la posibilidad de integrarlo con otros procesos en comparación con otras aplicaciones descarta la viabilidad de este trabajo actualmente.

A pesar de haber superado el principal problema que tenía la aplicación con la implantación del cifrado de extremo a extremo, *WhatsApp* siguió siendo vulnerable a ataques para interceptar la conversación entre dos usuarios.

Esta vulnerabilidad se basa en el uso del protocolo SS7. Este protocolo fue elaborado en 1975 y se definió como estándar en 1981. El principal propósito de este protocolo es el establecimiento y finalización de llamadas.

El ataque aprovecha la posibilidad de robar el SMS de validación de cuenta que WhatsApp envía a sus clientes para poder suplantar la identidad y así poder interceptar la comunicación.

Según WhatsApp, actualmente esta vulnerabilidad se ha corregido y ya no es posible interceptar la comunicación entre dos usuarios mediante este método.

2.1.1.2. Telegram:

Lanzado en enero de 2014, en un primer momento *Telegram* se constituyó como una aplicación de mensajería muy similar a *WhatsApp*, de código abierto y centrado desde el principio en ofrecer un mayor abanico de funcionalidades que su homologo.

Figura 2: Mensaje de presentación *Telegram*

Telegram se ha centrado desde el primer momento en incluir en los mensajes entre usuarios una mayor seguridad frente a la intrusión de terceros. *Telegram* permite crear grupos de hasta mil usuarios permite a sus usuarios enviar archivos de todo tipo y de mayor tamaño que los permitidos por *WhatsApp*, así como crear los llamados "grupos de difusión", una especie de "blog" en formato de mensajería instantánea.

La diferencia principal con el resto de alternativas es la existencia de bots. Estos bots permiten el envío de imágenes relacionadas con un tema en concreto, recibir avisos automáticos por el chat, jugar al trivial, consultar precios de Amazon… etc.

Telegram también permite que se ejecuten scripts a la vez que un cliente de la aplicación asociado a una cuenta de teléfono en un pc. Aprovechando esta funcionalidad desarrollaremos nuestro proyecto.

2.1.1.2.1. Carencias

Telegram cuenta con un número bastante inferior de usuarios que *WhatsApp.* En febrero de 2016 tuvo 100 millones de usuarios, 10 veces menos que *WhatsApp.*

A pesar de que desde su creación el factor diferenciador ha sido la seguridad que esta aplicación ofrecía, la vulnerabilidad que se ha explicado en el apartado **Carencias** de *WhatsApp* también afecta a esta aplicación.

2.1.2. Aplicaciones comida a domicilio – Competidores:

Se consideran como competidores aquellas aplicaciones que permiten realizar un pedido de comida a un restaurante desde un terminal móvil o una Tablet.

2.1.2.1. Just Eat:

La aplicación para realizar pedidos a domicilio de la empresa Just Eat fue lanzada para Android en agosto de 2013 y para IOs en noviembre del mismo año. Durante 2015 ha alcanzado 100 mil descargas al mes.

Figura 3: Logo JUST EAT

2.1.2.1.1. Funcionamiento:

La aplicación tiene un funcionamiento similar al de la página web. El usuario introduce su código postal y la aplicación le devuelve los restaurantes de la zona con la carta y las referencias de este. La aplicación permite la geolocalización del usuario, ofreciendo una alternativa a introducir el código postal a mano.

Una vez seleccionado el restaurante y los platos, el usuario puede introducir la dirección y el pago por cuenta bancaria o tarjeta de crédito en la misma aplicación.

2.1.2.1.2. Carencias:

La aplicación permite una comunicación muy limitada entre el encargado del restaurante y los clientes. Esto puede provocar que, si surge algún problema, una buena coordinación entre el cliente y el establecimiento sea difícil de alcanzar.

La interfaz en algunos casos puede ser algo caótica, probando la aplicación se encuentran casos en los que al realizar algunos pedidos la aplicación nos lleva por distintos menús sin dejar claro en qué punto de la elaboración del pedido estamos.

2.1.2.2. La nevera roja:

La aplicación de La Nevera roja se lanzó en mayo de 2013 tanto en Android como en IOs.

Figura 4: Ejemplo de la aplicación “La nevera roja”

2.1.2.2.1. Funcionamiento:

Tiene un funcionamiento similar a la aplicación de JU, de hecho, ambas empresas pertenecen al mismo grupo y tienen interfaces y metodología unificada.

La diferencia principal con JU es que LNR no requiere que introduzcas un código postal, sino una calle. Esto simplifica el proceso de compra para el cliente.

2.1.2.2.2. Carencias:

La experiencia con la aplicación es bastante mejor que con la aplicación de Just Eat, mas intuitiva y ágil, hasta llegar al apartado en donde se nos pide información de contacto. En nuestra opinión, sin embargo, pide más información de la necesaria para realizar un pedido a domicilio, por ejemplo, una dirección de correo electrónico.

2.1.2.3. Deliveroo:

Es una aplicación que permite realizar pedidos a diferentes restaurantes, independientemente de si tienen servicio de comida a domicilio. Para ello, cuentan con un equipo de repartidores que se encargan de llevar el pedido que el cliente ha realizado a su casa, además, permite que el cliente siga en tiempo real la ubicación de su pedido.

2.1.2.3.1. Funcionamiento:

La aplicación requiere que se introduzca una dirección o activar el gps del dispositivo como paso previo para mostrar los restaurantes a los que el cliente puede realizar el pedido.

Una vez localizado el lugar de entrega, el cliente puede navegar entre los distintos restaurantes, los cuales se identifican con su nombre y una foto representativa del tipo de comida que venden. Al elegir un restaurante, aparece la carta y tan solo hay que pulsar en los platos que queremos comprar para que se vayan añadiendo a la cesta.

Cuando el cliente ha terminado, debe pulsar un botón visible en todo momento para dar por finalizado el pedido.

El último paso es introducir las credenciales de usuario o registrarnos en el sistema. Cuando estamos identificados, debemos añadir un método de pago y una dirección en caso de que sea el primer pedido que realicemos o queramos cambiar alguno de estos.

2.1.2.3.2. Carencias:

El principal problema de la aplicación es que solo presta servicio en grandes poblaciones, por ejemplo, si intentamos ver los restaurantes introduciendo una dirección en la zona de la sierra noroeste, la aplicación nos informa que en esa zona todavía no realizan repartos.

2.2. Tecnología:

El script desarrollado funciona sobre un cliente de *Telegram* para Linux desarrollado por **Vhyseng**. Trabajar con un código desarrollado por otra persona nos obliga a trabajar con una tecnología determinada.

2.2.1. Sistema operativo:

Como hemos dicho, el cliente que se utilizará como base está desarrollado para funcionar en Linux.

Linux es un sistema operativo inspirado en MINIX, un sistema UNIX desarrollado por Andy Tanenbaum. Fue elaborado por Linus Torval como una afición, teniendo en mente a aquellos usuarios que necesitaban algo más que lo que ofrecía MINIX en ese momento. Durante 1991, el proyecto fue evolucionando hasta que en octubre de ese mismo año se lanzó la primera versión oficial de Linux.

Durante todos estos años este sistema operativo ha ido creciendo gracias a la colaboración de toda la comunidad de programadores. Linux es software libre, lo que significa que no es necesario pagar ninguna licencia para su utilización, además, su código fuente se puede consultar desde el propio sistema, esto permite a cualquier usuario (con los conocimientos necesarios) modificar el código fuente, reiniciar el equipo y tener su propio sistema operativo personalizado.

Gracias a la versatilidad del sistema, han surgido muchos proyectos para elaborar distintos sistemas operativos basados en Linux orientados a diferentes objetivos.

Según la orientación que le hayan dado los desarrolladores, existen distintas distribuciones. Estas distribuciones pueden estar centradas en la auditoria de redes, en conseguir mayor estabilidad, en ofrecer un entorno para los usuarios con un perfil menos técnico o en conseguir el mínimo consumo de recursos para funcionar en equipos de bajo rendimiento.

2.2.2. Telegram-CLI de Vysheng:

Telegram-CLI es una librería elaborada por Vysheng y alojada en GitHub. Como otros clientes de *Telegram* para PC, permite acceder a todas las funciones de *Telegram* sin necesidad de un terminal móvil, tan solo con validar el número a través del mismo cliente mediante un SMS que se envía previamente al número que introduzcamos, podremos utilizar la aplicación desde el PC.

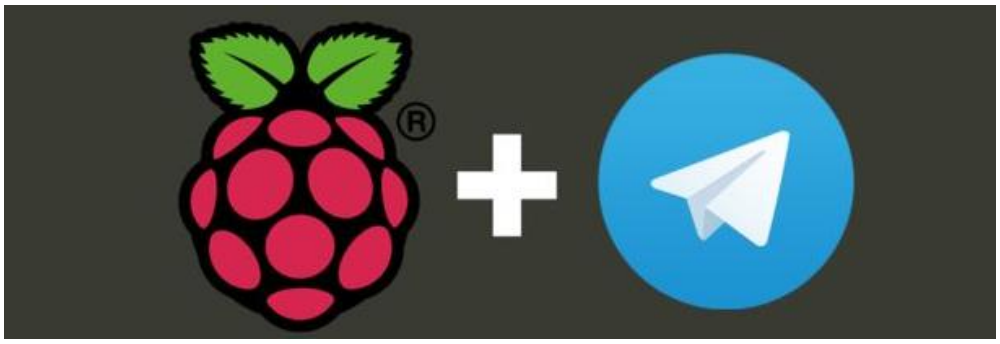


Figura 5: Ilustración de la integración de *Raspbian* y *Telegram*

La ventaja que ofrece este cliente es que permite ser ejecutado a la vez que un script, el cual ejecutará determinadas líneas de código en función de los eventos que lance el cliente.

Por ejemplo, podemos escribir un script para que cuando se reciba el mensaje “Hola” desde uno de nuestros contactos, se le conteste automáticamente con la cadena de texto “Hola, ¿qué tal?”

Gracias a la posibilidad de ejecutar comandos de la terminal desde el propio script, podemos crear carpetas, ejecutar programas, leer de ficheros, escribir en ellos o borrar archivos de la misma forma que un usuario podría hacerlo mediante la terminal de Linux.

Gracias a la completa libertad que este sistema brinda, podemos utilizar un gestor de bases de datos como MySQL junto a él para combinar la recepción de pedidos con el acceso a la información almacenada en una base de datos.

2.2.3. MySQL:

MySQL es el sistema de gestión de bases de datos relacionales más utilizado del mundo, fue desarrollado en 1995. Actualmente pertenece a Oracle, aunque sigue manteniendo la filosofía de código abierto que tenía cuando se creó.

Figura 6: Logo de MySQL

He decidido usar MySQL como método para gestionar una base de datos debido a la sencillez que ofrece, tiene un funcionamiento y unos comandos fáciles de utilizar, así como una gran cantidad de documentación.

Poder utilizar una interfaz gráfica como complemento a la interfaz por línea de comandos es una gran ventaja, además de los conocimientos previos de este lenguaje adquiridos durante la carrera.

El uso de esta herramienta es gratuito siempre que sea con fines no comerciales. A pesar de que se pretende comercializar el sistema desarrollado, de momento el desarrollo entrará en el uso de gratuito para proyectos no comerciales.

2.2.4. Lenguaje de programación:

No es necesario realizar cambios en el cliente, tan solo se implementarán las funcionalidades deseadas del script.

2.2.4.1. Script para el Cliente:

Ya que utilizamos la librería de Vysheng, no se pueden utilizar más lenguajes que LUA o Python.

El script será desarrollado en LUA. La decisión de utilizar este lenguaje viene motivada por el gran número de ejemplos de uso de este para el cliente de *Telegram* frente a la alternativa, Python.

2.2.5. Entorno de desarrollo:

He utilizado Sublime como procesador de textos. Esta decisión se debe a la comodidad y familiaridad que ofrece este programa. Ya que no es necesaria ninguna instalación de plugin o bibliotecas adicionales, toda la implementación se ha realizado en texto plano que luego era compilado y ejecutado a través de la terminal.

3. Análisis del sistema:

Con el objetivo de cumplir plazos y desarrollar todas las funcionalidades necesarias se especifica en este apartado todo lo que tendrá que tener en cuenta el equipo de desarrollo.

3.1. Definición del sistema:

Se analiza el alcance, restricciones y el entorno operacional del sistema para definirlo.

3.1.1. Alcance:

El alcance explica el problema que pretendemos solucionar mediante el sistema desarrollado.

El Script a implementar permite a los clientes de los establecimientos de comida a domicilio obtener información del establecimiento y del pedido que esté realizando, así como consultar el menú, los precios y ofertas. El cliente podrá elaborar el pedido mediante una conversación con el sistema instalado en el restaurante utilizando un lenguaje lo más natural posible.

Según el número de teléfono que interactúe con el sistema se podrán diferenciar distintos usuarios (clientes, repartidor, cocina, gerente). Cada usuario accederá a unos mensajes y comandos aceptados diferente, aunque puede darse el caso de acciones comunes entre usuarios. Esto permite que todos los aspectos de la gestión del pedido puedan ser controlados a través del sistema desarrollado.

3.1.2. Restricciones:

- El Script será desarrollado en el lenguaje de programación LUA.
- Se utilizará el repositorio de Vysheng como base para la implementación del sistema.
- MySQL se utilizará para gestionar la base de datos.

3.1.3. Entorno operacional:

- El Script se ejecutará en un Linux Ubuntu 14.06 para realizar el desarrollo y las pruebas.
- El sistema final funcionará sobre un Raspbian.
- Los usuarios solo necesitarán tener una versión funcional de la aplicación *Telegram*. A pesar de que cada usuario podrá realizar unas operaciones y consultas concretas, todas las acciones se realizaran a través de *Telegram*.

3.2. Especificación de requisitos:

Los requisitos especifican los criterios que han de ser usados para juzgar la operatividad y los comportamientos específicos de un sistema (Wiegers, 2003).
Podemos considerarlo como un contrato entre el cliente y los desarrolladores del sistema, que sirve como base para estimar los costes y los tiempos de desarrollo.

Se utilizará la siguiente plantilla para realizar la definición de cada requisito:

Identificador	Valor único para cada requisito. Formado por dos partes, la referencia al grupo del requisito (UF para requisitos funcionales y NF para los requisitos no funcionales) y el número de este dentro de su grupo.
Titulo	Describe brevemente en que consiste el requisito.
Descripción	Breve explicación del requisito.
Prioridad	• Alta: El requisito tiene gran importancia para que el sistema cumpla las funcionalidades deseadas. • Media: El requisito es importante, pero no crítico. • Baja: El requisito será implementado en último lugar
Necesidad	• Crítico: El requisito es esencial, sin su implementación el sistema no cumple todas las funcionalidades necesarias. • Deseable: Mejora el sistema, pero no es imprescindible. • Opcional: La implementación no es necesaria para el correcto funcionamiento.
Verificabilidad	• Alta: Es sencillo verificar el cumplimiento del requisito. • Baja: La verificación es compleja.
Estabilidad	• Alta: El requisito no es modificable durante el tiempo. • Media: El requisito es modificable si cambian los requerimientos. • Baja: El requisito es muy susceptible de sufrir cambios durante la vida del sistema.

Tabla 2: Patrón de tabla especificación de requisitos

3.2.1. Requisitos de usuario:

Describen los requerimientos de forma comprensible para el cliente, aunque no posea un conocimiento técnico detallado.

Especifican únicamente que comportamiento deberá tener el sistema mediante un lenguaje natural, evitando en la medida de lo posible aspectos del diseño del sistema.

Tenemos dos tipos, requisitos funcionales y requisitos no funcionales.

- Funcionales: Establecen los comportamientos del sistema.
- No funcionales: Establecen el diseño o la implementación.

3.2.1.1. Requisitos funcionales:

Identificador	UF01
Titulo	Consulta de la carta del establecimiento
Descripción	El sistema debe permitir a cada usuario acceder a la carta
Prioridad	Alta
Necesidad	Crítico
Verificabilidad	Alta
Estabilidad	Media

Tabla 3: UF01

Identificador	UF02
Titulo	Selección entre “Recoger” el pedido o “A domicilio
Descripción	El sistema debe registrar donde se recibirá el pedido
Prioridad	Alta
Necesidad	Crítico
Verificabilidad	Alta
Estabilidad	Alta

Tabla 4: UF02

Identificador	UF03
Titulo	Añadir platos al pedido
Descripción	El sistema debe permitir añadir platos al pedido.
Prioridad	Alta
Necesidad	Crítico
Verificabilidad	Alta
Estabilidad	Alta

Tabla 5: UF03

Identificador	UF04
Titulo	Eliminar platos del pedido
Descripción	El sistema debe permitir eliminar platos del pedido.
Prioridad	Alta
Necesidad	Crítico
Verificabilidad	Alta
Estabilidad	Alta

Tabla 6: UF04

Identificador	UF05
Titulo	Consultar los platos del pedido.
Descripción	El sistema debe poder mostrar el contenido del pedido al usuario si este lo requiere.
Prioridad	Alta
Necesidad	Crítico
Verificabilidad	Alta
Estabilidad	Alta

Tabla 7: UF05

Identificador	UF06
Titulo	Validar el pedido.
Descripción	El usuario debe poder cerrar el pedido cuando haya terminado las operaciones sobre él
Prioridad	Alta
Necesidad	Crítico
Verificabilidad	Alta
Estabilidad	Alta

Tabla 8: UF06

Identificador	UF07
Titulo	Consultar estado del pedido.
Descripción	El usuario podrá conocer el estado del pedido. “En espera”, “Cocina”, “En camino”
Prioridad	Alta
Necesidad	Crítico
Verificabilidad	Alta
Estabilidad	Alta

Tabla 9: UF07

Identificador	UF08
Titulo	Modificar el estado del pedido.
Descripción	El encargado podrá modificar el estado del pedido. “En espera”, “Cocina”, “En camino”
Prioridad	Alta
Necesidad	Crítico
Verificabilidad	Alta
Estabilidad	Alta

Tabla 10: UF08

Identificador	UF09
Titulo	Consultar información del pedido.
Descripción	El encargado podrá consultar la información del pedido. Dirección, Teléfono, Importe.
Prioridad	Alta
Necesidad	Crítico
Verificabilidad	Alta
Estabilidad	Alta

Tabla 11: UF09

Identificador	UF10
Titulo	Consultar identificador y hora del pedido.
Descripción	El encargado podrá consultar los identificadores y la hora a la que se realizó cada pedido.
Prioridad	Alta
Necesidad	Crítico
Verificabilidad	Alta
Estabilidad	Alta

Tabla 12: UF10

Identificador	UF11
Titulo	Recibir una notificación al producirse un nuevo pedido
Descripción	El sistema avisará con un mensaje de la creación de un nuevo pedido.
Prioridad	Alta
Necesidad	Media
Verificabilidad	Alta

Estabilidad	Alta

Tabla 13: UF11

Identificador	UF12
Titulo	Marcar pedido como “Entregado”
Descripción	El usuario determinado (Gerente o Repartidor) podrá indicar al sistema que un pedido determinado ha sido entregado y así permitir al cliente realizar un nuevo pedido cuando así lo considere.
Prioridad	Alta
Necesidad	Media
Verificabilidad	Alta
Estabilidad	Alta

Tabla 14: UF12

Identificador	UF013
Titulo	Consulta de los ingredientes de un producto
Descripción	El sistema debe permitir a cada usuario acceder a los ingredientes del producto seleccionado
Prioridad	Alta
Necesidad	Crítico
Verificabilidad	Alta
Estabilidad	Media

Tabla 15: UF013

3.2.1.2. Requisitos no funcionales:

Identificador	UN01
Titulo	Telegram.
Descripción	El usuario solo puede comunicarse con el sistema mediante la aplicación Telegram.
Prioridad	Alta
Necesidad	Crítico
Verificabilidad	Alta
Estabilidad	Alta

Tabla 16: UN01

Identificador	UN02
Titulo	Acceso Internet usuarios.
Descripción	El usuario debe estar conectado a Internet para comunicarse con el sistema.
Prioridad	Alta
Necesidad	Crítico
Verificabilidad	Alta
Estabilidad	Alta

Tabla 17: UN02

Identificador	UN03
Titulo	Conexión a Internet del sistema.
Descripción	El sistema debe estar conectado a Internet para poder prestar servicio.
Prioridad	Alta
Necesidad	Crítico
Verificabilidad	Alta

Estabilidad	Alta

Tabla 18: UN03

Identificador	UN04
Titulo	Lenguaje.
Descripción	El usuario debe conocer el idioma en el que se enviaran los mensajes, generalmente, Castellano.
Prioridad	Alta
Necesidad	Crítico
Verificabilidad	Alta
Estabilidad	Media

Tabla 19: UN04

3.3. Casos de uso:

Describen la interacción del usuario con el sistema. Están relacionados con los requisitos funcionales, definiendo los actores involucrados y las condiciones previas y posteriores necesarias para llevar a cabo el requisito.

Al igual que con los requisitos funcionales, se utiliza una plantilla para describir los casos de uso.

Caso de uso	Valor único para cada caso de uso, lo identifica inequívocamente.

Actor	Todos aquellos que pueden iniciar una conversación con otro actor, independientemente de la necesidad de recibir un mensaje de respuesta, cada uno accede a una interfaz determinada (los mensajes que puede aceptarle el sistema y con los mensajes que este responderá)
	• Personal del establecimiento
	  • Cocina
	  • Repartidor
	  • Gerente
	• Cliente
	• Sistema

Objetivo	Motivo por el que se genera el caso de uso
Condiciones Previas	Condiciones necesarias previamente para la ejecución

Condiciones Posteriores	Condiciones necesarias a posteriori para la ejecución
Requisitos funcionales relacionados	Requisitos funcionales que tienen relación con el caso de uso

Tabla 20: Patrón de tabla de casos de uso

Los casos de uso identificados en el sistema son los siguientes:

Caso de uso	Consultar carta del establecimiento
Actor	Cliente
Objetivo	El cliente debe poder consultar la carta del establecimiento, para introducir los nombres de los platos que desea
Condiciones Previas	El cliente debe tener un pedido abierto
Condiciones Posteriores	El cliente tiene en su dispositivo un mensaje enumerando los platos de la carta
Requisitos funcionales relacionados	UF01

Tabla 21: Caso de uso Consultar carta del establecimiento

Caso de uso	Creación de pedido
Actor	Cliente
Objetivo	El cliente debe poder crear un pedido asociado a él para registrar los platos y la dirección en donde desea recibirlo
Condiciones Previas	El cliente no debe tener un pedido abierto
Condiciones Posteriores	Existe un pedido “abierto” en la base de datos asociado al número del cliente.
Requisitos funcionales relacionados	UF02

Tabla 22: Caso de uso Creación de pedido

Caso de uso	Modificar los platos del pedido
Actor	Cliente
Objetivo	El cliente debe poder añadir o eliminar platos del pedido en curso
Condiciones Previas	El cliente debe tener un pedido abierto
Condiciones Posteriores	El cliente recibe un mensaje informando del plato que se ha añadido o el listado de los platos del pedido después de terminar el proceso de eliminación de platos
Requisitos funcionales relacionados	UF03, UF04

Tabla 23: Caso de uso Modificar los platos del pedido

Caso de uso	Consultar los datos del pedido
Actor	Cliente
Objetivo	El cliente debe poder ver en todo momento que platos tiene anotados en su pedido
Condiciones Previas	El cliente debe tener un pedido abierto
Condiciones Posteriores	El cliente recibe un mensaje con el listado de los platos que contiene su pedido
Requisitos funcionales relacionados	UF05

Tabla 24: Caso de uso Consultar los datos del pedido

Caso de uso	Cerrar el pedido
Actor	Cliente
Objetivo	El cliente debe poder indicar al sistema que ha terminado de añadir platos a su pedido
Condiciones Previas	El cliente debe tener un pedido abierto
Condiciones Posteriores	El cliente recibe un mensaje informando de que su pedido se ha cerrado y con los platos que este contiene
Requisitos funcionales relacionados	UF06

Tabla 25: Caso de uso Cerrar el pedido

Caso de uso	Consultar estado del pedido
Actor	Cliente
Objetivo	El cliente debe poder consultar en qué estado se encuentra su pedido
Condiciones Previas	El cliente debe tener un pedido abierto y marcado como finalizado
Condiciones Posteriores	El cliente recibe un mensaje informando del estado en el que se encuentra su pedido
Requisitos funcionales relacionados	UF07

Tabla 26: Caso de uso Consultar estado del pedido

Caso de uso	Consultar los ingredientes de un pedido
Actor	Cliente
Objetivo	Durante la elaboración del pedido, el cliente debe poder consultar los ingredientes de cada plato
Condiciones Previas	El cliente debe tener un pedido abierto y marcado como finalizado
Condiciones Posteriores	El gerente recibe un mensaje indicando que un nuevo pedido se ha marcado como finalizado
Requisitos funcionales relacionados	UF13

Tabla 27: Caso de uso Consultar los ingredientes de un pedido

Caso de uso	Modificar el estado del pedido
Actor	Personal del establecimiento
Objetivo	Según el punto de la elaboración y entrega del pedido, el gerente debe poder indicar al sistema el estado correspondiente.
Condiciones Previas	El cliente debe tener un pedido abierto y marcado como finalizado
Condiciones Posteriores	El gerente recibe un mensaje indicando a qué estado se ha cambiado el pedido
Requisitos funcionales relacionados	UF08

Tabla 28: Caso de uso Modificar el estado del pedido

Caso de uso	Consultar la información de cada pedido
Actor	Personal del establecimiento
Objetivo	El gerente debe poder acceder a datos como la dirección, numero o importe de cada pedido

Condiciones Previas	El cliente debe tener un pedido abierto y marcado como finalizado
Condiciones Posteriores	El gerente recibe un mensaje con la información del pedido escogido
Requisitos funcionales relacionados	UF09

Tabla 29: Caso de uso Consultar la información de cada pedido

Caso de uso	Consultar hora e identificador de cada pedido
Actor	Personal del establecimiento
Objetivo	El personal del establecimiento podrá consultar la identificación dentro del sistema de los pedidos que se encuentran pendientes, así como la hora a la que se realizaron.
Condiciones Previas	El cliente debe tener un pedido abierto y marcado como finalizado
Condiciones Posteriores	El gerente recibe un mensaje con la información del pedido escogido
Requisitos funcionales relacionados	UF10

Tabla 30: Caso de uso Consultar hora e identificador de cada pedido

Caso de uso	Aviso de nuevo pedido
Actor	Sistema
Objetivo	Al cerrarse un nuevo pedido, el sistema tiene que tener la capacidad de notificar al gerente correspondiente de este evento.
Condiciones Previas	El cliente debe tener un pedido abierto y marcado como finalizado
Condiciones Posteriores	El gerente recibe un mensaje indicando que un nuevo pedido se ha marcado como finalizado
Requisitos funcionales relacionados	UF11

Tabla 31: Caso de uso Aviso de nuevo pedido

Caso de uso	Marcar pedido como entregado
Actor	Personal del establecimiento
Objetivo	Al realizar la entrega y el pago correspondiente por parte del cliente, el repartidor o el gerente
Condiciones Previas	El cliente debe tener un pedido abierto y marcado como finalizado
Condiciones Posteriores	El gerente recibe un mensaje indicando que un nuevo pedido se ha marcado como finalizado
Requisitos funcionales relacionados	UF12

Tabla 32: Caso de uso Marcar pedido como entregado

3.3.1. Diagrama de casos de uso:

3.3.1.1. Cliente:

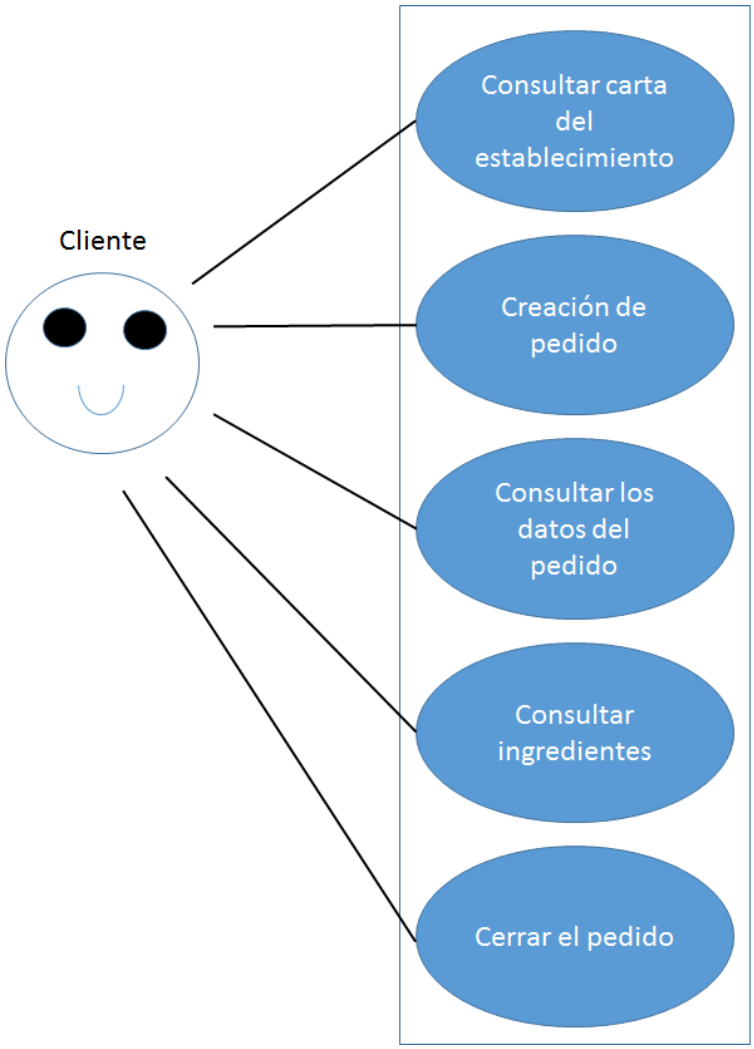


Figura 7: Diagrama de casos de uso del cliente

3.3.1.2. *Personal establecimiento y sistema:*

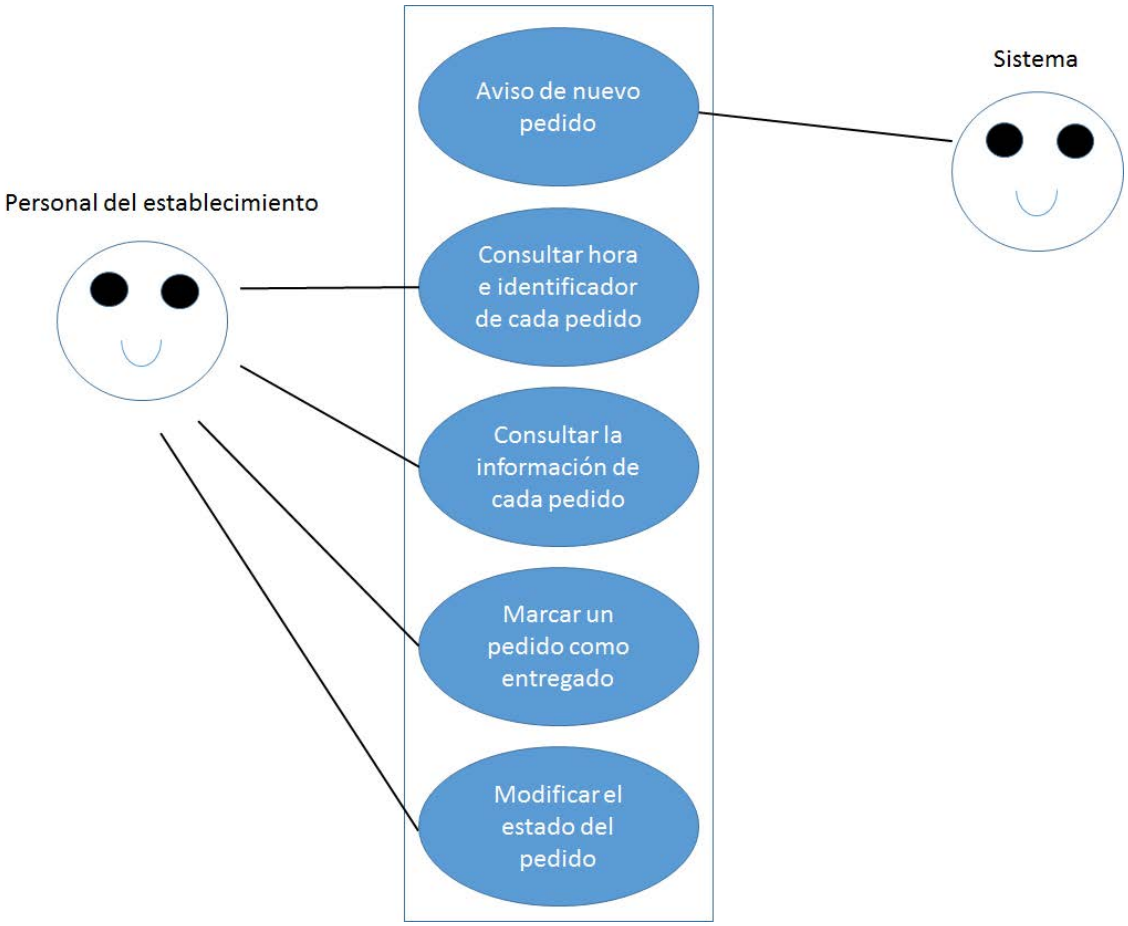


Figura 8: Diagrama de casos de uso personal del establecimiento y del sistema

3.4. Requisitos del software:

Los requisitos de software especifican que es lo que va a ser implementado. El IEEE 830 define que una buena especificación de requisitos de software debe ser (Piattini, 1996) y (Chalmeta, 2000):

- Correcta
- No ambigua
- Completa
- Verificable
- Consistente
- Clasificada
- Modificable
- Explorable
- Utilizable tanto en tareas de mantenimiento como en el uso

Estos requisitos se pueden clasificar en dos tipos según se refieran a funcionalidades del sistema (**Requisitos funcionales**) o bien restricciones que debe tener el software (**Requisitos no funcionales**). (Sommerville, 2004).

La plantilla que se utilizará para definir estos requisitos será la misma que la utilizada para definir los requisitos de usuario. La identificación será **RSF** o **RSNF** para los requisitos funcionales o para los requisitos no funcionales respectivamente seguido del número de requisito en cada caso.

3.4.1. Requisitos funcionales:

Identificador	RSF01
Titulo	Consulta de la carta del establecimiento
Descripción	El sistema debe permitir a cada usuario acceder a la carta. El usuario enviará un mensaje con “Carta” para consultar el tipo de platos que se ofertan. Una vez conocidos que tipo de platos se ofertan, el usuario introducirá en el que esté interesado para ver las distintas variaciones de estos. Es decir, si introduce “Hamburguesa” el sistema contestará con las distintas hamburguesas de la carta.
Prioridad	Alta
Necesidad	Crítico
Verificabilidad	Alta
Estabilidad	Media

Tabla 33: RSF01

Identificador	RSF02
Titulo	Crear pedido a domicilio y elegir lugar de recogida
Descripción	En función del mensaje enviado por el usuario, el campo de la base de datos “Recoger” se rellenará con una cadena de texto igual a “Domicilio” o “Recoger”.
Prioridad	Alta
Necesidad	Crítico
Verificabilidad	Alta
Estabilidad	Alta

Tabla 34: RSF02

Identificador	RSF03
Titulo	Añadir platos al pedido.
Descripción	El sistema añadirá los platos introducidos por el usuario en la tabla del pedido correspondiente. El sistema acepta números antes del plato para indicar cantidades.
Prioridad	Alta
Necesidad	Crítico
Verificabilidad	Alta
Estabilidad	Alta

Tabla 35: RSF03

Identificador	RSF04
Titulo	Eliminar platos del pedido
Descripción	Si el usuario introduce “eliminar”. los platos introducidos después del comando serán eliminados de la tabla del pedido en vez de añadidos. El sistema acepta números antes del plato para indicar cantidades.

Prioridad	Alta
Necesidad	Crítico
Verificabilidad	Alta
Estabilidad	Alta

Tabla 36: RSF04

Identificador	RSF05
Titulo	Consultar los platos del pedido.
Descripción	Durante la elaboración del pedido, el usuario podrá introducir en todo momento el texto “Listar”. El sistema le devolverá un mensaje con los platos que lleva registrados hasta el momento.
Prioridad	Alta
Necesidad	Crítico
Verificabilidad	Alta
Estabilidad	Alta

Tabla 37: RSF05

Identificador	RSF06
Titulo	Validar el pedido.
Descripción	El usuario podrá introducir “Fin” en todo momento durante la elaboración del pedido, al recibir ese mensaje, el sistema terminará la fase de elaboración y marcará el pedido en la base de datos como terminado. En la columna de “Estado” dentro de la tabla del pedido se introducirá el texto “En espera” para que los cocineros sepan que el pedido está pendiente de elaboración.
Prioridad	Alta
Necesidad	Crítico
Verificabilidad	Alta

Estabilidad	Alta

Tabla 38: RSF06

Identificador	RSF07
Titulo	Consultar estado del pedido.
Descripción	Una vez dado por terminado el pedido, el usuario podrá introducir “Estado” para saber en qué punto del proceso se encuentra el pedido. El sistema consultará la columna “Estado” del pedido correspondiente y le devolverá la información al usuario.
Prioridad	Alta
Necesidad	Crítico
Verificabilidad	Alta
Estabilidad	Alta

Tabla 39: RSF07

Identificador	RSF08
Titulo	Modificar el estado del pedido.
Descripción	Los dispositivos de los empleados podrán modificar el valor del campo “Estado” mediante un mensaje con la identificación del pedido y el estado al que debe cambiar.
Prioridad	Alta
Necesidad	Crítico
Verificabilidad	Alta
Estabilidad	Alta

Tabla 40: RSF08

Identificador	RSF9
Titulo	Consultar listado de pedidos pendientes.
Descripción	El encargado podrá consultar el listado de pedidos pendientes. Obtendrá un mensaje

	con el identificador de cada pedido, la hora a la que se ha cerrado y el importe de este.
Prioridad	Alta
Necesidad	Crítico
Verificabilidad	Alta
Estabilidad	Alta

Tabla 41: RSF9

Identificador	RSF10
Titulo	Consultar información del pedido.
Descripción	El encargado podrá consultar la información del pedido. Dirección, Teléfono, Importe. Para ello introducirán el identificador del pedido seguido de una “L”.
Prioridad	Alta
Necesidad	Crítico
Verificabilidad	Alta
Estabilidad	Alta

Tabla 42: RSF10

Identificador	RSF11
Titulo	Recibir una notificación al producirse un nuevo pedido.
Descripción	Cuando un pedido se dé por cerrado, el sistema enviará un mensaje al número de teléfono (indicado en la base de datos) del empleado correspondiente, notificándole la existencia de un nuevo pedido.
Prioridad	Alta
Necesidad	Media
Verificabilidad	Alta

Estabilidad	Alta

Tabla 43: RSF11

Identificador	RSF12
Titulo	Marcar pedido como “Entregado”.
Descripción	El usuario determinado (Gerente o Repartidor) podrá indicar al sistema que un pedido determinado ha sido entregado y así permitir al cliente realizar un nuevo pedido cuando así lo considere.
Prioridad	Alta
Necesidad	Media
Verificabilidad	Alta
Estabilidad	Alta

Tabla 44: RSF12

Identificador	UF013
Titulo	Consulta de los ingredientes de un producto.
Descripción	El sistema escribirá de vuelta un listado de los ingredientes de un producto. Esta acción será requerida por el usuario mediante la introducción del nombre del plato acompañado de un comando identificado.
Prioridad	Alta
Necesidad	Crítico
Verificabilidad	Alta
Estabilidad	Media

Tabla 45: UF013

3.4.2. Requisitos No funcionales:

3.4.2.1. Tolerancia a fallos:

Identificador	RSNF01
Titulo	Guiado para el usuario.
Descripción	Ante un mensaje no esperado, el sistema responderá con un mensaje explicando las posibles elecciones que puede hacer el usuario y cuál es el mensaje para cada uno.
Prioridad	Alta
Necesidad	Critica
Verificabilidad	Alta
Estabilidad	Alta

Tabla 46: RSNF01

Identificador	RSNF02
Titulo	Respuesta continua.
Descripción	El sistema siempre contestará con un mensaje a los usuarios que le escriban. Tanto para confirmar que todo va bien como para informar de que se ha producido cualquier error.
Prioridad	Alta
Necesidad	Critica
Verificabilidad	Alta
Estabilidad	Alta

Tabla 47: RSNF02

Identificador	RSNF03
Titulo	Comprobación de existencia de platos introducidos.
Descripción	Antes de que el sistema realice la inserción de un nuevo plato en la tabla del pedido correspondiente comprobará que el plato introducido existe en la base de datos del restaurante.
Prioridad	Alta
Necesidad	Crítico
Verificabilidad	Alta
Estabilidad	Alta

Tabla 48: RSNF03

Identificador	RSNF04
Titulo	Usuarios.
Descripción	El sistema debe ser capaz de distinguir entre los distintos usuarios posibles del sistema. Esta diferenciación vendrá dada por el número de teléfono del dispositivo que envíe el mensaje.
Prioridad	Alta
Necesidad	Crítico
Verificabilidad	Alta
Estabilidad	Alta

Tabla 49: RSNF04

3.4.2.2. Operación:

Identificador	RSNF05
Titulo	Conexión a internet usuarios.
Descripción	Es necesario que los usuarios tengan acceso a internet en sus dispositivos .
Prioridad	Alta
Necesidad	Crítico
Verificabilidad	Alta
Estabilidad	Alta

Tabla 50: RSNF05

Identificador	RSNF06
Titulo	Conexión a internet sistema.
Descripción	Es necesario que el sistema cuente con conexión a internet para dar servicio a los usuarios.
Prioridad	Alta
Necesidad	Crítico
Verificabilidad	Alta
Estabilidad	Alta

Tabla 51: RSNF06

3.4.2.3. Plataforma

Identificador	RSNF07
Titulo	Telegram.
Descripción	El sistema modificará y accederá a la base de datos mediante la terminal. Los comandos que la terminal ejecutará vendrán dados por los mensajes recibidos por el cliente de *Telegram* que corre en el equipo.

Prioridad	Alta
Necesidad	Crítico
Verificabilidad	Alta
Estabilidad	Alta

Tabla 52: RSNF07

Identificador	RSNF08
Titulo	Lenguaje.
Descripción	Los mensajes que recibirán los usuarios estarán elaborados en Castellano. El idioma podrá modificarse en caso de que el cliente así lo desee.
Prioridad	Alta
Necesidad	Crítico
Verificabilidad	Alta
Estabilidad	Media

Tabla 53: RSNF08

3.4.2.4. Rendimiento:

Identificador	RSNF09
Titulo	Tiempo de respuesta.
Descripción	Las respuestas que dé el sistema no deberán demorarse más de 1 segundo con una velocidad de conexión móvil de 4g y una velocidad de ADSL de 5Mg.
Prioridad	Media
Necesidad	Deseable
Verificabilidad	Alta
Estabilidad	Media

Tabla 54: RSNF09

Identificador	RSNF10
Titulo	Carga del sistema.
Descripción	El cliente no debe ocupar más de un 80% del trabajo del procesador y de la memoria para prevenir que el sistema se congele.
Prioridad	Media
Necesidad	Critica
Verificabilidad	Alta
Estabilidad	Media

Tabla 55: RSNF10

3.4.2.5. Interfaz:

Identificador	RSNF11
Titulo	Mensajes coherentes.
Descripción	Los mensajes deben tener una coherencia en el orden en el que el usuario los recibe.
Prioridad	Media
Necesidad	Crítico
Verificabilidad	Alta
Estabilidad	Alta

Tabla 56: RSNF11

Identificador	RSNF12
Titulo	Diferenciación de usuarios.
Descripción	Cada tipo de usuario debe de ser distinguido por el sistema para poder adaptar las respuestas y los mensajes que espera de él.
Prioridad	Alta
Necesidad	Crítico

Verificabilidad	Alta
Estabilidad	Alta

Tabla 57: RSNF12

Identificador	RSNF13
Titulo	Mensajes claros y concretos.
Descripción	Los mensajes emitidos por el sistema deben de estar lo más sintetizados posible.
Prioridad	Alta
Necesidad	Crítico
Verificabilidad	Media
Estabilidad	Alta

Tabla 58: RSNF13

3.4.2.6. Documentación y mantenibilidad:

Identificador	RSNF14
Titulo	Documentación.
Descripción	Los comentarios y la documentación del código serán en castellano.
Prioridad	Media
Necesidad	Deseable
Verificabilidad	Alta
Estabilidad	Alta

Tabla 59: RSNF14

Identificador	RSNF15
Titulo	Nombrar variables.
Descripción	Las variables se nombrarán indicando su función en el código de la forma más clara posible

Prioridad	Media
Necesidad	Deseable
Verificabilidad	Alta
Estabilidad	Alta

Tabla 60: RSNF15

Identificador	RSNF16
Titulo	Versiones.
Descripción	Las versiones del código se almacenarán localmente y se diferenciarán mediante el nombre del archivo, indicando fecha y los comentarios adicionales necesarios al principio del código.
Prioridad	Media
Necesidad	Deseable
Verificabilidad	Alta
Estabilidad	Alta

Tabla 61: RSNF16

3.4.2.7. Seguridad y privacidad

Identificador	RSNF17
Titulo	Cifrado de información sensible
Descripción	Los datos de los consumidores estarán cifrados dentro de la base de datos
Prioridad	Alta
Necesidad	Critica
Verificabilidad	Alta
Estabilidad	Alta

Tabla 62: RSNF17

3.4.2.8. Testabilidad:

Identificador	RSNF18
Titulo	El código debe ser testable.
Descripción	Se desarrollará el código de forma que sea lo más fácil de testear posible.
Prioridad	Baja
Necesidad	Deseable
Verificabilidad	Baja
Estabilidad	Baja

Tabla 63: RSNF18

3.5. Matriz de trazabilidad

Mediante la matriz de trazabilidad podemos observar como hay al menos un requisito de software por cada requisito de usuario.

	RSF01	RSF02	RSF03	RSF04	RSF05	RSF06	RSF07	RSF08	RSF09	RSF10	RSF11	RSF12	RSF13	RSNF07	RSNF05	RSNF06	RSNF08
UF01	■																
UF02		■															
UF03			■														
UF04				■													
UF05					■												
UF06						■											
UF07							■										
UF08								■									
UF09									■								
UF10										■							
UF11											■						
UF12												■					
UF13													■				
UN01														■			
UN02															■		
UN03																■	
UN04																	■

Tabla 64: Matriz de trazabilidad

4. Diseño:

Este apartado explica el diseño que se ha tomado tanto de la interfaz que presentará el sistema para los distintos usuarios, así como diseño de los procesos que pueden producirse durante el uso de este.

Se ha dividido la explicación en dos partes, la parte visible para el usuario y la parte interna del sistema (Instalación y uso del cliente de *Telegram* y la base de datos utilizada).

El sistema operativo se configurará para que al iniciarse se ejecute el cliente junto al script para que el personal del establecimiento tan solo tenga que esperar a esta ejecución si fuera necesario reiniciar el equipo.

4.1. Funcionamiento interno:

La parte interna consta de tres partes:

1. **Cliente de Telegram:**

Este programa se encarga de la recepción y la emisión de mensajes hacia los servidores de *Telegram.* Ha sido implementado por un desarrollador ajeno y proviene de un repositorio de GitHub. No se han realizado modificaciones en el código del programa.

El cliente permite interactuar con los contactos del usuario que registremos al iniciar el programa mediante la terminal de Linux, se pueden realizar las mismas acciones que si estuviéramos conectados mediante la aplicación móvil.

2. **Script:**

Es el código que hemos implementado, representa todas las funcionalidades del sistema aparte de las ofrecidas por el cliente. Se ejecuta junto al cliente y trabaja conjuntamente, el cliente se encarga de gestionar la comunicación con Telegram y el script se encarga de interactuar con el sistema operativo, con la base de datos y de indicarle al cliente que mensajes debe enviar y cuando.

Permite elaborar una lógica para que se cree la sensación de una conversación con el interlocutor, que este interactúe con la información del establecimiento y pueda realizar cambios en el estado del sistema.

3. Base de datos:

La base de datos dará soporte al funcionamiento del script. Debido a la naturaleza del script, ante un nuevo mensaje, se volverá a lanzar su ejecución, lo que impide que sea posible que guardemos datos entre mensajes consecutivos. Esto haría que sin base de datos fuese imposible conservar el estado de la ejecución (punto de la conversación, platos indicados, respuestas encadenadas).

En un principio, se tomó como solución a esta problemática el uso de ficheros de texto plano para ir guardando la información de los clientes según se iba generando, así como los platos que tenía el establecimiento y toda la información de los empleados. Este método, sencillo de utilizar debido a la facilidad para crear, modificar o eliminar ficheros mediante el uso de llamadas a la terminal de Linux fue abandonado por lo poco escalable que es, los problemas que generaba diseñar el formato de la información contenida en los ficheros, así como lo farragoso de generar ficheros cada vez que un nuevo usuario generaba un pedido o interactuaba con el sistema. La versión final utiliza la base de datos para guardar la información necesaria para el correcto funcionamiento del sistema.

El uso de una base de datos MySQL permite organizar la información de una forma ordenada, que el proceso de añadir información sea sencillo y que las consultas que realicemos sean fáciles de parsear para utilizarlas dentro del código.

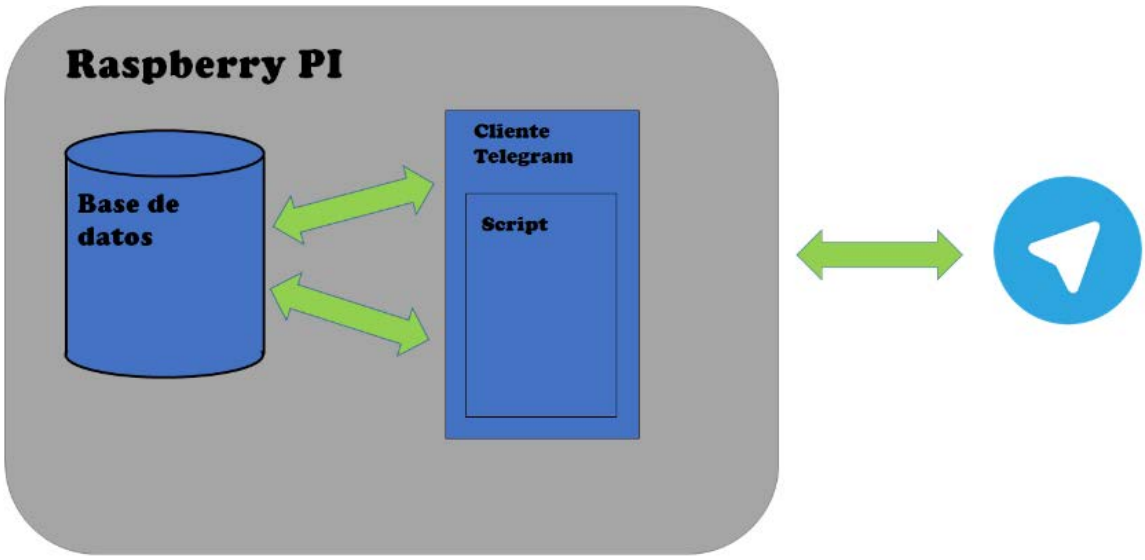


Figura 9: Diseño simplificado del sistema

4.1.1. Instalación y ejecución de Telegram-Cli de Vysheng

Todos los pasos que se han seguido han sido obtenidos de la página de GitHub de Vysheng (https://github.com/vysheng/tg).

El primer paso es copiar el repositorio en nuestro equipo, esto se realiza mediante el comando `git`, utilizamos la opción `--recursive`. Omitir esta opción produciría errores durante el proceso.

```
git clone --recursive https://github.com/vysheng/tg.git && cd tg
```

Figura 10: Clonación repositorio de GitHub

La ejecución del comando anterior copiará todo el repositorio en una carpeta llamada "tg" dentro de la carpeta en la que nos encontremos.

Una vez copiado el repositorio, procedemos a la instalación de las librerías que serán necesarias para que el cliente funcione. Ya que estamos trabajando en un equipo Ubuntu, utilizaremos el comando `apt-get install` con privilegios de usuario root.

```
sudo apt-get install libreadline-dev libconfig-dev libssl-dev lua5.2
liblua5.2-dev libevent-dev libjansson-dev libpython-dev make
```

Figura 11: Instalación librerías necesarias para el funcionamiento del cliente

Por último, ejecutamos el archivo "configure" y realizamos un `make` para compilar todo.

```
./configure
make
```

Figura 12: Ejecución y compilación

Una vez realizados estos pasos, ya podemos ejecutar el cliente. Esta ejecución permite el uso del cliente como sustituto de la aplicación móvil sin ningún añadido, es decir, podremos recibir y escribir mensajes de la misma forma que mediante la aplicación, pero a través de la consola de nuestro equipo. Aparte de este método de ejecución, el cliente puede ser ejecutado junto a un script.

```
bin/telegram-cli -k tg-server.pub -W -s action.lua
```

Figura 13: Ejecución del cliente junto a un script

Este modo de ejecución nos permite indicarle al cliente que acciones debe tomar ante determinados eventos, por ejemplo, recibir un mensaje de un número determinado, contestar en función del mensaje recibido, etc.

4.1.2. Script

En este apartado se explica el funcionamiento del script desarrollado.

El script viene con funciones predefinidas que deberemos modificar en función de las acciones que queremos que realice el cliente. El método que debemos modificar es `on_msg_receive` ya que es el que se consulta al recibir un nuevo mensaje. Aparte de esta función, existe la posibilidad de implementar acciones concretas para chats "secretos"[1] por ejemplo.

El cliente tiene más funcionalidades de las utilizadas, sin embargo, la documentación existente en el repositorio del cliente de *Telegram* es bastante escasa, por lo que la implementación ha tratado de ensayo y error, manteniéndose en los límites de lo necesario para conseguir que el sistema tenga las funcionalidades deseadas.

El script implementado en el repositorio viene con un pequeño ejemplo para ilustrar su funcionamiento, gracias a él se puede empezar a entender cómo funciona el flujo de este código.

Gracias a este ejemplo se pudo comenzar a comprender como funciona el cliente y a elaborar las primeras versiones de lo que acabaría siendo el resultado final del sistema elaborado.

[1] Estos chats no permiten que los mensajes escritos en ellos se reenvíen a otros chats, además, si alguno de los interlocutores elimina alguno de los mensajes, este queda eliminado en todos los dispositivos del chat.

Este script indica al cliente que cuando reciba un mensaje de cualquier número con el texto “ping” debe contestar a ese mismo número “pong”.

```
function on_msg_receive (msg)
          if started == 0 then
            return
          end
          if msg.out then
            return
          end
          do_notify (get_title (msg.from, msg.to), msg.text)


          if (msg.text == 'ping') then
                send_msg (msg.from.print_name, 'pong', ok_cb, false)
        return
                end

        end
```

Figura 14: Ejemplo de script

Para acceder a la información del emisor del mensaje que se ha recibido, utilizamos la variable `msg`. Entre los atributos que contiene, encontramos `msg.from.phone`, que nos devolverá el número de teléfono desde el que nos han escrito el último mensaje y `msg.from.print_name`, que nos da el nombre de usuario de Telegram del emisor del mensaje.

Utilizaremos el número de teléfono para almacenar la información de contacto del cliente y para que los encargados del establecimiento puedan llamarle en caso de que sea necesario.

El nombre de usuario en *Telegram* es importante para permitir que el sistema mande mensajes automáticamente a un dispositivo concreto. Para evitar que se pueda utilizar el cliente junto a un script para mandar spam automáticamente a un gran número de teléfonos, es necesario conocer el nombre de usuario del dispositivo al que queremos escribir el mensaje. Si quisiéramos que el sistema, por ejemplo, notificase al encargado de la cocina de la aparición de un nuevo pedido.

```
send_msg ('Cocina_restaurante', 'Hay un nuevo pedido, escribe h para
          ver que pedidos están pendientes', ok_cb, false)
```

Figura 15: Ejemplo de implementación de mensaje automático

Esta línea tendría que estar justo después del código que da como finalizado el pedido y lo añade al listado de pedidos pendientes.

En el momento de dar de alta el dispositivo en *Telegram* sería necesario utilizar 'cocina_restaurante' como nombre de usuario cuando se nos pida que introduzcamos uno

Ya que LUA permite la ejecución de órdenes por terminal desde el código, podremos modificar la acción a realizar para que cree carpetas, archivos, ejecute programas, etc.

4.1.2.1. Funciones implementadas:

Aparte de las funciones predefinidas el funcionamiento del script permite que creemos nuestras propias funciones según las necesidades que tengamos.

Para indicar al sistema que vamos a declarar una nueva función, hay que seguir un patrón muy sencillo. La primera palabra debe ser `function`, esto indica que estamos ante el inicio de una función, después, se debe poner el nombre que queramos que tenga la función para luego poder invocarla desde otra parte del código seguido de los atributos que recibirá al ser invocada. Estos atributos están delimitados por paréntesis y separados por comas. Para marcar el fin del código de la función, usamos la palabra reservada `end`.

Para realizar las modificaciones en las tablas, obtenemos el id de la cuenta que tiene asociada el número de teléfono desde el que nos están escribiendo.

```
--Introducir el comando SQL para la busqueda del id de cuenta de la cuenta
        file = io.open("temp_borrar.txt", "w")
        io.output(file)
        io.write("use restaurante;select id from Cuenta where Telefono='",telefono,"';")
        io.close (file)
```

Figura 16: Ejemplo de ejecución de comando desde el script

En función de la acción que queremos realizar sobre la base de datos, habrá que operar con la salida de la ejecución. En los casos en los que se realiza una consulta, habrá que recortar el texto de esta salida para obtener el dato que queremos.

Para facilitar el uso por parte de los usuarios del sistema, todos los mensajes recibidos se transforman a mayúsculas antes de su análisis, de esta forma, evitamos forzar a los usuarios que escriban siempre de una forma determinada o ampliar el código para contemplar todas las formas que podrían utilizar para escribir sus mensajes (trabajo que sería inabarcable si se pretendiese contemplar absolutamente todas las formas).

En la mayoría de las funciones implementadas, se envía como parámetros el atributo msg, esto se debe a que mediante esta variable podemos acceder a toda la información del emisor del mensaje, así como el contenido de este, es decir, toda la información que podríamos necesitar.

4.1.2.1.1. Función on_msg_receive:

La declaración de esta función viene predefinida en el script que viene con el código del repositorio, sin embargo, representa el núcleo de todo el sistema. Se encarga de procesar cada mensaje que el cliente recibe y decidir qué acción debe tomar el sistema.

El código de esta función puede ser divido en varias partes:

- **Comprobación del número de teléfono:**

Al recibir un nuevo mensaje, lo primero que se hace es comprobar si el número de teléfono que ha escrito el mensaje está registrado en alguna de las tablas de la base de datos que contienen la información de los empleados. En caso positivo, el flujo del script se modifica para que no reciba los mensajes que recibiría un cliente del restaurante y pueda acceder a la parte reservada para ellos.

Si el número de teléfono no está registrado como parte del personal, el sistema considerará que se trata de un cliente y ejecutará la parte reservada para este caso.

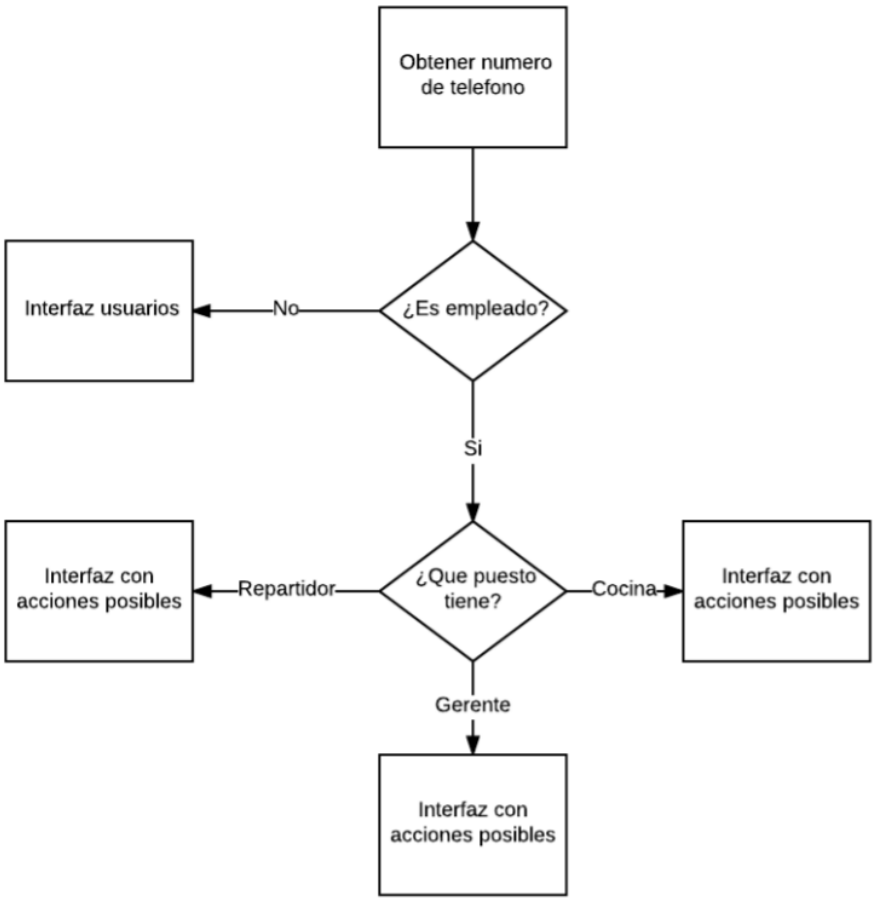


Figura 17: Diagrama de flujo Distinción de usuarios

- **Acciones vinculadas al tipo de usuario:**

En esta parte, se le indican al usuario los comandos que puede introducir y que resultados tendrá el uso de cada uno de ellos.

Cuando el sistema no encuentra ninguna concordancia en la base de datos entre el número de teléfono del dispositivo que ha escrito el mensaje y los que tiene almacenados, contesta con un texto de bienvenida e indicaciones para comenzar a elaborar el pedido al primer mensaje que reciba en caso de que no sea un comando reconocido. Esto se debe a que, si el cliente es capaz de enviar mensajes que el sistema puede reconocer como válidos para elaborar un pedido o suministrar información, es que ya conoce como interactuar con el sistema y tampoco necesita un mensaje de bienvenida.

Cuando el cliente ya ha escrito uno o más mensajes, el mensaje de bienvenida no aparecerá, aunque siempre recibirá un mensaje aclarando que puede hacer en el caso de que escriba cualquier cosa que el sistema no sea capaz de reconocer como comando válido. En todo momento se ha intentado que los mensajes estén escritos en un lenguaje natural y de la forma más clara posible.

Si el usuario forma parte de los empleados, recibe un mensaje informando de el efecto que ha tenido el mensaje que ha introducido o un listado de los mensajes que puede enviar y que efectos tendrán estos en el caso de que el último mensaje no se haya reconocido.

En el caso de que el gerente del establecimiento desee que el código este configurado para que cada tipo de empleado pueda realizar unas acciones determinadas, será en esta parte donde se realizará la distinción de una forma similar a la que se realiza con los clientes y el personal.

En esta función es desde la que se realizan la mayoría de las llamadas al resto de las funciones implementadas.

4.1.2.1.2. Función extraer:

```
function extraer(id,valor_extraer,tabla)
        local file = io.open("temp_extraer.txt", "w")
        io.output(file)
        io.write("use restaurante;select ",valor_extraer," from ",tabla," where id=",id,";")
        io.close (file)
  --Guardar los nombres de los productos
        local handle = io.popen("mysql -u pedidos < temp_extraer.txt")
        local resultados = handle:read("*a")
        handle:close()
        --print("resultados"..resultados)
        return resultados
end
```

Figura 18: Ejemplo de función implementada

Esta función recibe el identificador de una cuenta, el valor que queremos consultar dentro de la base de datos y la tabla en donde se encuentra ese valor. Es un buen ejemplo de las posibilidades que nos ofrece LUA, se puede observar que permite la apertura de ficheros mediante la llamada `io.open` o ejecutar comandos de la misma forma que en la consola de Linux mediante el comando `io.popen`. Entraremos más en detalle sobre el uso de llamadas al sistema operativo (a partir de ahora SO) desde el script durante este apartado.

Se ha elaborado esta función debido a la cantidad de veces que se tenían que realizar consultas de distintos atributos de las tablas de la base de datos con el mismo id. Este método pretende recortar la longitud del código que utilizamos, para conseguir una mayor calidad y facilidad de depuración.

4.1.2.1.3. Función borrar:

Esta función se encarga de eliminar platos del pedido que el cliente está elaborando. Recibe tan solo como atributo la variable `msg`.

Se la invocará en caso de que el cliente introduzca borrar como primera palabra en un mensaje, después se obtendrá el resto del mensaje para comprobar que plato quiere eliminar, la cantidad que desea eliminar y si es posible realizar la acción. Esta llamada se realiza desde la función `on_msg_receive`.

```
local recorte=string.upper(msg.text)
recorte=string.sub(recorte,1,6)
if(recorte=='BORRAR') then
  borrar(msg)
```

Figura 19: Ejemplo de llamada a una función

El usuario puede introducir un numero para indicar cuantos productos quiere eliminar, de esta forma puede evitar mandar varios mensajes para eliminar un mismo plato. En caso de que introduzca más platos para eliminar que los que tiene eliminados, el sistema le mandará un mensaje indicando que no puede realizarse la operación de eliminar platos.

Recibirá otro mensaje informándole de que no ha introducido ningún plato si el texto que hay después de “borrar” no corresponde a ningún plato o no contiene texto.

```
--Comprobar si se ha introducido algun producto despues del numero
    if(producto=='') then
      send_msg (msg.from.print_name, 'No ha introducido ningun plato para borrar', ok_cb, false)
    else
```

Figura 20: Ejemplo de mensaje de respuesta al interlocutor

Si el usuario no introduce ningún número, el sistema interpretará que desea eliminar una sola unidad del plato elegido.

Si el plato y las unidades que ha introducido son correctas, el sistema eliminará de la base de datos tantas tablas “Cuenta_producto” como se haya indicado en la cantidad, para ello seleccionará aquellas entradas en las que el atributo “nombre_producto” y “id_cuenta” concuerden con el producto introducido y con el id de la cuenta vinculada al número de teléfono.

```
--Modificar la entrada de Cuenta_producto para anotar los productos seleccionados
  --Introducir comando SQL
        file = io.open("temp_borrar.txt", "w")
        io.output(file)
        io.write("use restaurante;delete from Cuenta_producto where nombre_producto='",producto,"' and id_cuenta=",id_cuenta," limit ",cantidad,";")
        io.close (file)
  --Ejecutar el comando sql
        handle = io.popen("mysql -u pedidos < temp_borrar.txt")
        result = handle:read("*a")
        io.close (handle)
        send_msg (msg.from.print_name, 'Se han eliminado '..cantidad..' '..producto, ok_cb, false)
```

Figura 21: Ejemplo de escritura en fichero y ejecución de comando

4.1.2.1.4. Función anotar_pedidos:

Esta función será invocada cuando se introduzca un numero antes del texto o bien si no se ha introducido ninguno de los comandos esperados (“listar”, “borrar”, “carta” o alguno de los nombres de los tipos de platos). Tan solo será accedida por los clientes del establecimiento.

Recibe como atributos de entrada el texto que indica el producto seleccionado, la cantidad que el usuario ha indicado que desea del producto, así como la variable `msg`. Esta función será llamada desde la función `on_msg_receive`.

La cantidad que se envía al método en caso de que el usuario no introduzca ninguna explícitamente es igual a uno.

Si el texto que se ha introducido, aparte de no corresponder con ninguno de los comandos tampoco lo hace con ninguno de los platos almacenados en la base de datos, el sistema informa al cliente de la situación, para ello, comprueba si la consulta del atributo “Precio” en la tabla de “Productos” con el texto introducido como atributo “nombre” produce algún resultado.

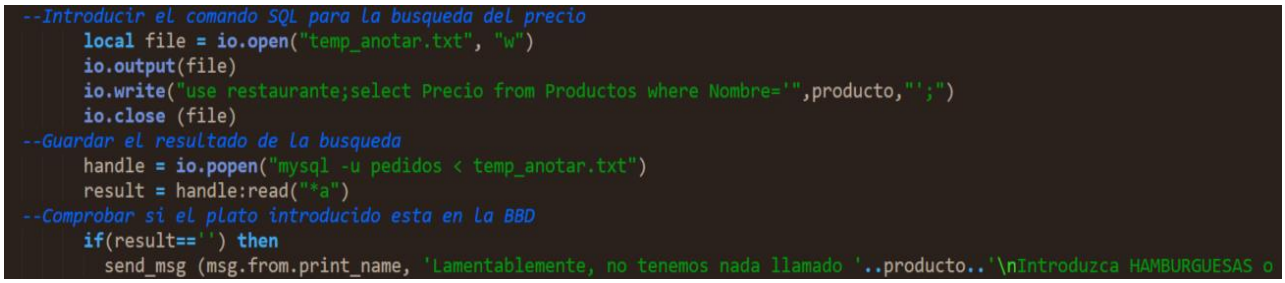


Figura 22: Ejemplo de consulta a la base de datos y mensaje al interlocutor

En caso de que la consulta produzca un resultado en la base de datos, el sistema informa al cliente del plato que se ha añadido al pedido y la cantidad de este plato que se ha anotado. Además, se calcula el importe de los platos añadidos.

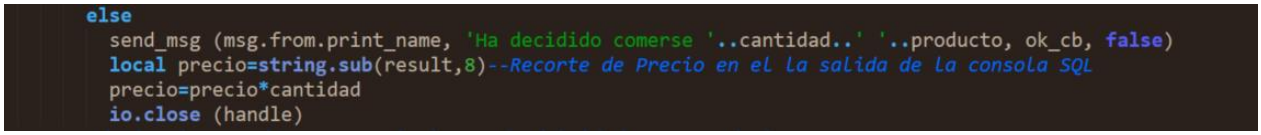


Figura 23: Ejemplo de mensaje al interlocutor y actualización de datos

Para actualizar el importe de la cuenta, se obtiene el antiguo valor del atributo “Precio” de la tabla de “Cuenta” correspondiente. El valor de este atributo se pone a 0 al crear la cuenta.

Una vez actualizado el importe de la cuenta, se hace un “UPDATE” (modificación de los datos de una tabla creada con anterioridad) sobre la tabla correspondiente y se almacena el valor calculado previamente. Para almacenar los productos añadidos, se crean tantas entradas de la tabla “Cuenta_producto” como productos se hayan introducido en el último mensaje del cliente.

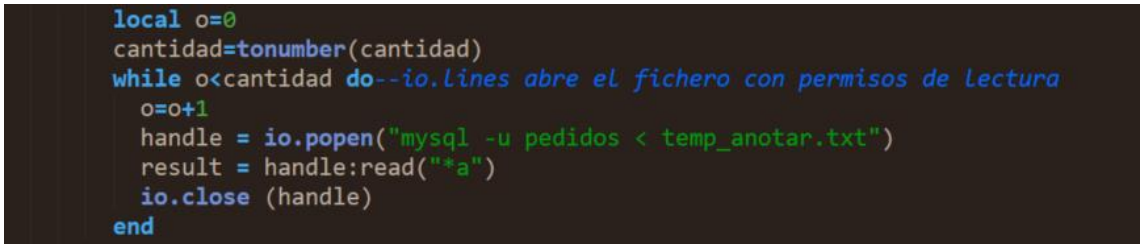


Figura 24: Ejemplo de ejecución de comando en bucle

4.1.2.1.5. Función listar:

Esta función se invoca cuando el usuario escribe un mensaje al sistema con el texto “listar”. La respuesta que recibe es un mensaje con un listado del nombre de los productos de la cuenta identificada por uno de los atributos enviados y el importe total de la cuenta.

Esta función puede ser accedida tanto por clientes del establecimiento como por los empleados del mismo. En el caso de que sea el cliente quien lo solicita, solo podrá acceder a la información de su propia cuenta, esto se consigue enviando por defecto el identificador asociado al número de teléfono que ha enviado el mensaje con la solicitud, si quien envía el mensaje forma parte del personal del restaurante, este deberá haber indicado en su mensaje el identificador de la cuenta que desea consultar. En el caso de que en este último caso el interlocutor no haya indicado ninguna cuenta, el sistema le mandará un mensaje informando del error.

En ambos casos, será llamada desde `on_msg_receive`.

Esta función recibe como parámetros el valor que identifica la cuenta de la que se ha solicitado el listado de los platos que contiene y la variable `msg`.

No se ha añadido el importe de cada uno de los productos debido a la complejidad que añadiría a la implementación por cómo se ha diseñado la organización de la base de datos. Se ha considerado que el usuario podrá consultar rápidamente el coste de los productos consultando la carta del establecimiento, acción que está disponible en todo momento de la elaboración del pedido.

4.1.2.1.6. Función borrar_pedido:

Esta función puede ser solo invocada por parte de los usuarios que pertenezcan al personal del establecimiento y cuenten con un dispositivo que tenga su número de teléfono registrado en el sistema. Será accedida desde la función `on_msg_receive`.

Alternativamente, el sistema también invocará este mensaje justo después de marcar un pedido concreto como “Entregado”.

Esta función se encarga de copiar la información de la tabla de un pedido que estaba activo, es decir, pendiente de entregar, en la tabla reservada en la base de datos para conservar el historial de los pedidos, “Cuenta_fin”.

Recibe como atributos el identificador de la cuenta que se desea eliminar y la variable `msg`. Cuando se realiza la operación, envía un mensaje al emisor del mensaje previo informándole de que se han realizado las acciones pertinentes.

4.1.3. Diseño de la base de datos.

En caso de que MySQL no esté instalado en la distribución de Linux que vayamos a usar para soportar el cliente, habrá que realizar la instalación.

Una vez realizada la instalación, hay que crear las tablas de la base de datos que servirán para almacenar toda la información necesaria.

Para realizar la creación de las tablas, usamos MySQL Workstation, una interfaz gráfica que simplifica el trabajo. La alternativa a esta opción sería realizar este trabajo mediante comandos escritos en la terminal de Linux, sin embargo, no es recomendable debido a la dificultad añadida que implica sin ofrecer ningún tipo de ventaja.

Una vez creadas las tablas, procedemos a utilizarlas para incluir en la base de datos la información del restaurante. Hemos utilizado un menú hipotético de un restaurante para realizar las pruebas.

Para representar la relación que hay entre las distintas tablas de la base de datos, utilizamos el modelo entidad-relación.

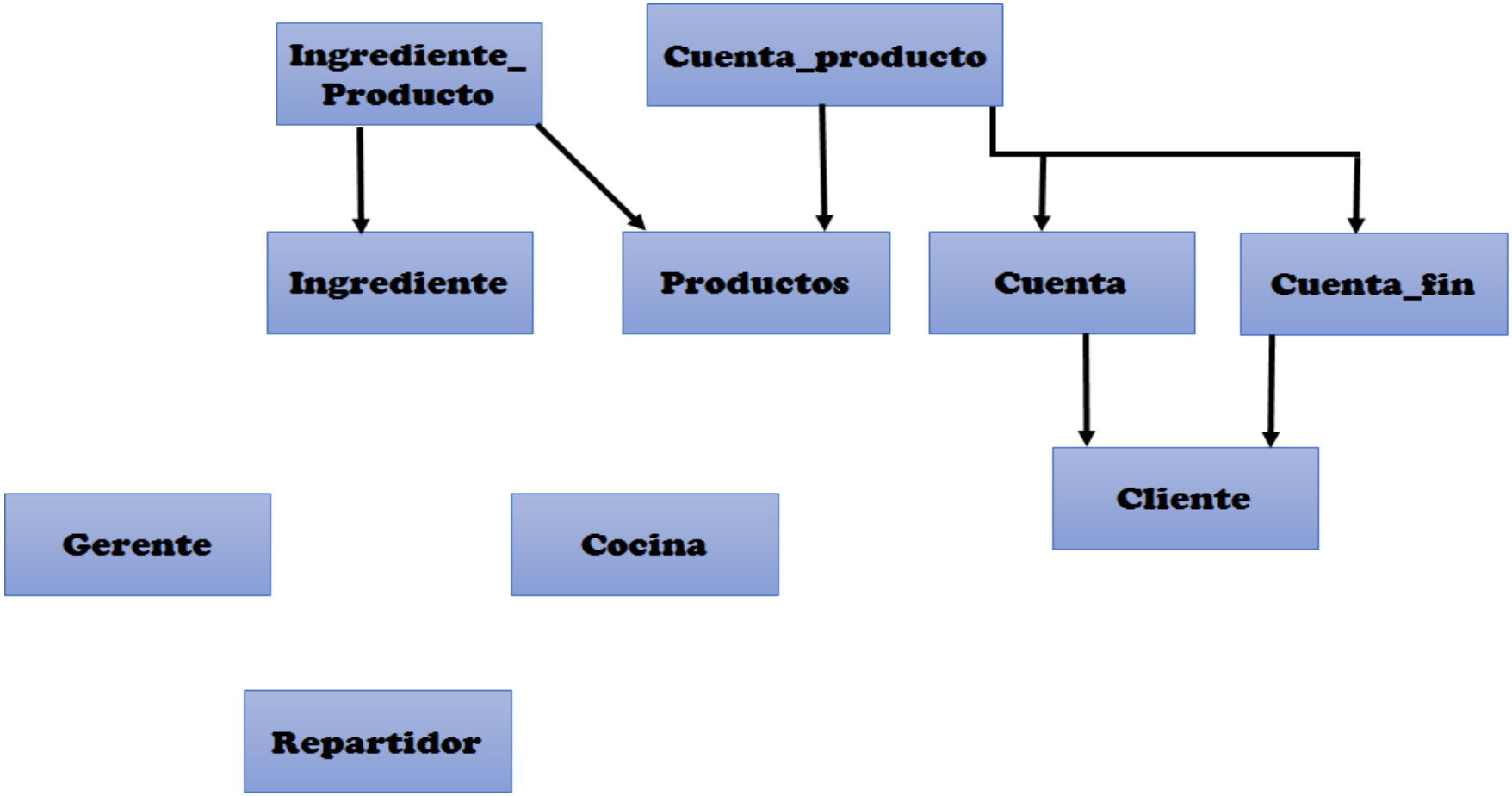


Figura 25: Diagrama Entidad-Relación

4.1.4. Tablas información productos:

Nombre de la tabla	Descripción
Productos	Tablas para almacenar que productos se venden en el establecimiento
id	Valor identificativo único. Integer
Nombre	Nombre común del producto. Cadena de caracteres
Precio	Importe del producto. Decimal
Tipo_producto	Clasificación del producto (Bebida, Acompañante, etc.). Cadena de caracteres

Tabla 65: Descripción de la tabla Productos

Nombre de la tabla	Descripción
Ingrediente	Tablas para almacenar los ingredientes utilizados en los productos ofertados
id	Valor identificativo único. Integer
Nombre	Nombre común del ingrediente Cadena de caracteres

Tabla 66: Descripción de la tabla Ingrediente

Nombre de la tabla	Descripción
Ingrediente_Producto	Tablas para vincular los ingredientes con los productos ofertados
id	Valor identificativo único. Integer
Nombre_ingrediente	Nombre común del ingrediente Cadena de caracteres
Nombre_producto	Nombre común del producto. Cadena de caracteres

Tabla 67: Descripción de la tabla Ingrediente_Producto

4.1.5. Tablas información pedidos:

Nombre de la tabla	Descripción
Cliente	Identificación del cliente
id	Valor identificativo único. Integer
Dirección	Cadena de caracteres
Teléfono	Cadena de caracteres

Tabla 68: Descripción de la tabla Cliente

Nombre de la tabla	Descripción
Cuenta	Pedido actual del cliente
id	Valor identificativo único. Integer
id_cliente	Valor identificativo de la tabla "Cliente" que corresponda
Teléfono	Cadena de caracteres
Precio	Valor actualizado del importe de la cuenta. Decimal
Dirección	Cadena de caracteres
entrega	Cadena de caracteres
cerrada	Cadena de caracteres
estado	Cadena de caracteres
fecha	Datos con formato fecha
hora	Datos con formato hora

Tabla 69: Descripción de la tabla Cuenta

Nombre de la tabla	Descripción
Cuenta_fin	Pedidos marcados como finalizados. Historial de pedidos
id	Valor identificativo único. Integer
id_cliente	Valor identificativo de la tabla "Cliente" que corresponda
Teléfono	Cadena de caracteres
Precio	Valor actualizado del importe de la cuenta. Decimal
Dirección	Cadena de caracteres
entrega	Cadena de caracteres
estado	Cadena de caracteres
fecha	Datos con formato fecha
hora	Datos con formato hora

Tabla 70: Descripción de la tabla Cuenta_fin

Nombre de la tabla	Descripción
Cuenta_producto	Tabla intermedia para vincular Productos a la Cuenta
nombre_producto	Nombre del producto
id_cuenta	Valor identificativo de la Cuenta

Tabla 71: Descripción de la tabla Cuenta_producto

4.1.6. Tabla información del personal del establecimiento:

Nombre de la tabla	Descripción
Gerente	Tablas para almacenar la información de los gerentes
id	Valor identificativo único. Integer
teléfono	Teléfono del dispositivo del gerente

Tabla 72: Descripción de la tabla Gerente

Nombre de la tabla	Descripción
Cocina	Tablas para almacenar la información de los dispositivos de Cocina
id	Valor identificativo único. Integer
teléfono	Teléfono del dispositivo de Cocina

Tabla 73: Descripción de la tabla Cocina

Nombre de la tabla	Descripción
Repartidor	Tablas para almacenar la información de los dispositivos de los repartidores
id	Valor identificativo único. Integer
teléfono	Teléfono del dispositivo del repartidor

Tabla 74: Descripción de la tabla Repartidor

4.2. Interfaz de los usuarios:

Todos los usuarios interactuaran con el sistema a través del chat de *Telegram*. Cada usuario podrá realizar unas acciones determinadas y recibirá unos mensajes distintos. Todos los mensajes son enviados y recibidos de la misma forma.

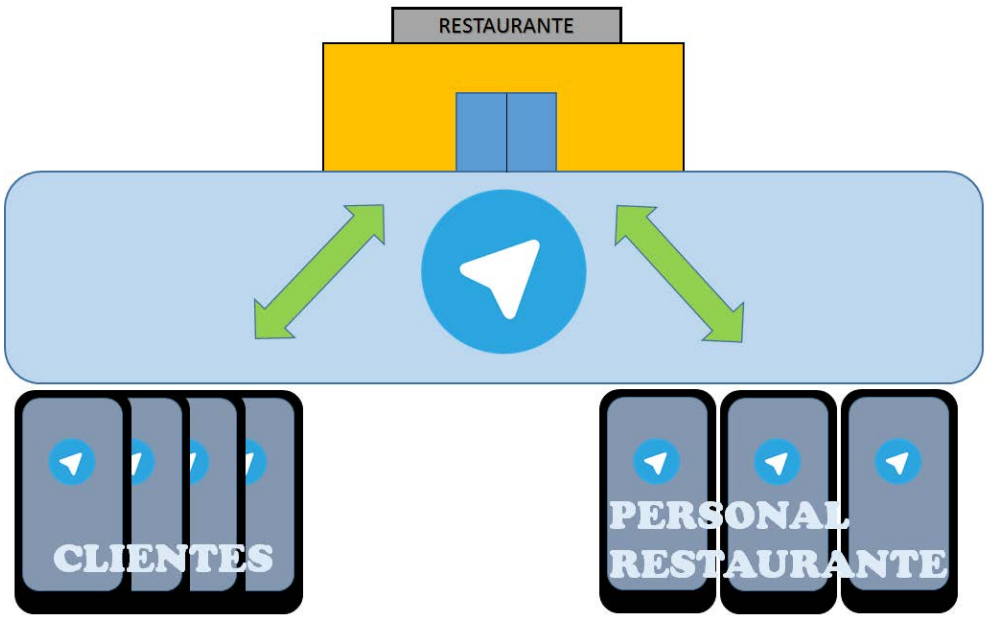


Figura 26: Ilustración de la interacción entre usuarios y el sistema a través de *Telegram*

Cuando el sistema recibe un nuevo mensaje, comprueba a que grupo de usuarios pertenece el móvil que ha escrito el mensaje. Para ello, se realiza una consulta a la base de datos.

La consulta envía el número de teléfono para recibir el grupo al que pertenece. Esta respuesta puede ser “gestor”, “cocina”, “repartidor” en función de la tabla en la que se encuentre alguna concordancia o bien informar de que la consulta no generó ningún resultado, lo que significará que el usuario es un cliente normal del establecimiento.

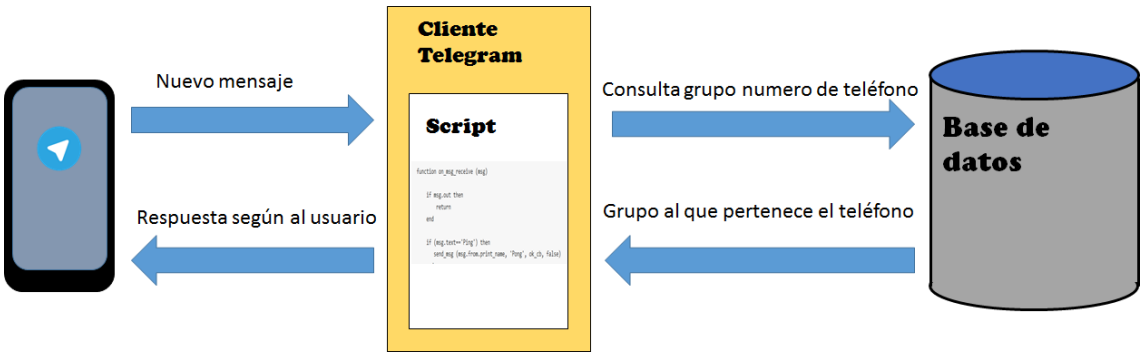


Figura 27: Proceso para procesar y responder a un nuevo mensaje

Cada usuario tiene unos privilegios establecidos previamente. Estos marcan los mensajes que se aceptaran y recibirán para cada nuevo número que escriba al sistema.

4.2.1. Personal del restaurante:

Si el número de teléfono pertenece a algún miembro del restaurante podrá estar identificado como uno de los siguientes usuarios.

4.2.1.1. Gerente:

Este usuario es el máximo responsable del establecimiento. Debe encargarse de que todo marche correctamente, por lo tanto, tiene capacidad para acceder a toda la información de la base de datos, así como modificar cualquier pedido o eliminarlo para que se vuelque la información en el histórico de pedidos. Esta eliminación debería realizarse automáticamente al marcar el pedido como “entregado” pero el gerente podrá realizarla manualmente para evitar que debido a un error un usuario no pueda realizar nuevos pedidos una vez que haya recibido el último que pidió.

GERENTE
Consultar referencia de pedidos pendientes ✓ Consultar información de un pedido ✓ Marcar el pedido como entregado ✓ Borrar pedidos ✓ Cambiar estado del pedido ✓

Tabla 75: Ejemplo de permisos para el usuario Gerente

4.2.1.2. Repartidor:

Este usuario solo tendrá que informar al sistema de que ha entregado el pedido para que se dé por terminado el proceso. Para conseguir la máxima agilidad en el proceso de entrega, podrá acceder a la información del pedido que elija (dirección, teléfono, platos e importe). El gerente solo le tendría que facilitar el numero identificador del pedido que tiene entregar y el repartidor se encargará de comprobar que el pedido está en orden antes de salir a entregarlo, así como enterarse de donde tiene que llevarlo. Una vez en el domicilio, podrá consultar el importe del mismo para saber cuánto tiene que cobrar al cliente.

REPARTIDOR
Marcar el pedido como entregado ✓ Consultar información de un pedido ✓

Tabla 76: Ejemplo de permisos para el usuario Repartidor

4.2.1.3. Cocina:

Este usuario es el encargado de elaborar los pedidos que luego serán servidos en el establecimiento. Es necesario que pueda consultar la referencia de los pedidos que están pendientes, así como acceder a la información concreta de cada uno de ellos para poder saber que platos tendrán que preparar. Una vez que estén preparando el pedido serán los encargados de marcarlo como en preparación para que el cliente tenga la información de su pedido lo más actualizada posible, también le indicarán al sistema cuando el pedido está terminado y está a la espera de ser entregado.

COCINA
Cambiar estado del pedido ✓
Consultar información de un pedido ✓
Consultar referencia de pedidos pendientes ✓

Tabla 77: Ejemplo de permisos para el usuario Cocina

Este esquema es tan solo un ejemplo de cómo podría estar configurado el sistema, se podrá modificar el número de miembros del personal del restaurante o los permisos que tendrá cada uno de ellos.

El número de miembros de cada grupo vendrá dado por los requerimientos del restaurante. La única restricción es que cada dispositivo tendrá que pertenecer a un solo grupo. Esta restricción viene motivada por la necesidad de que los permisos y los usuarios estén organizados de una forma correcta. Al igual que en la interfaz del cliente, si el sistema no reconoce el mensaje que se le ha enviado, contestará con un mensaje explicando que comandos puede aceptar en ese momento y que respuesta se producirá con cada uno de ellos.

4.2.1.4. Ejemplos de uso:

A diferencia de la interfaz que se le presenta a los clientes del restaurante, los mensajes que utilizaran los empleados del establecimiento se han diseñado con la idea de que sean lo más rápido de escribir, aunque no estén en un lenguaje natural. Por ejemplo, para consultar el listado de pedidos que están pendientes de elaborar, su identificador y la hora a la que se ha realizado el pedido, tendrán que escribir tan solo una “h”.

Al escribir un mensaje que no espera el sistema, este contesta de la siguiente forma:

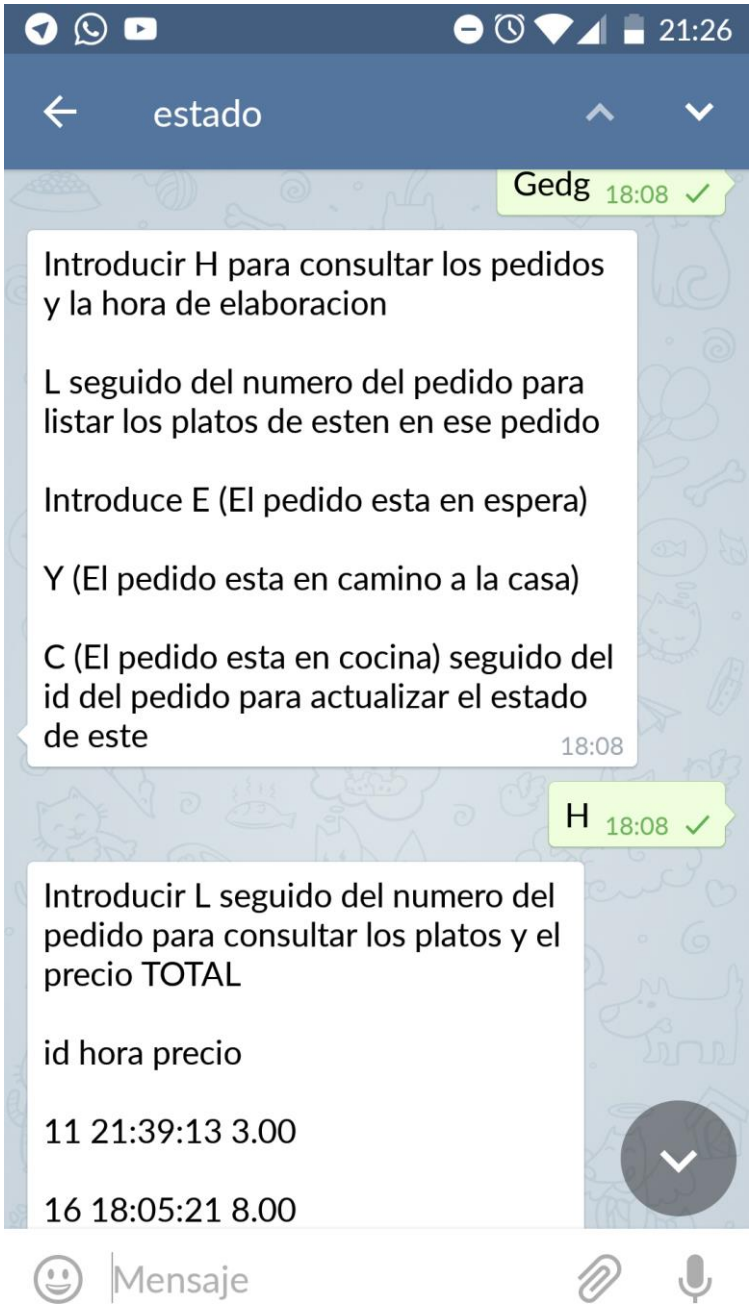


Figura 28: Pantalla de comandos del usuario Gerente

Para acceder a una información más detallada de cada pedido, es necesario el identificador que el sistema devuelve al escribir “h”

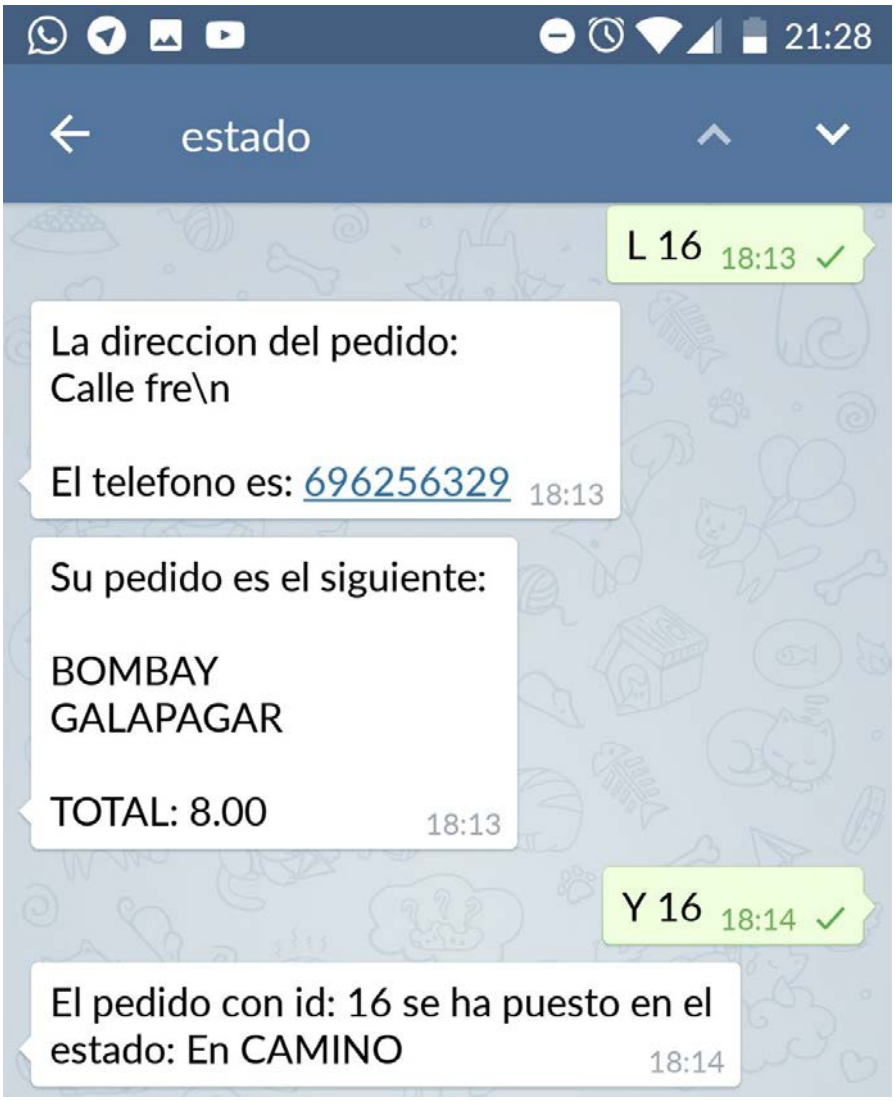


Figura 29: Pantalla de consulta de información concreta de un pedido

Con este identificador también se podrá modificar el estado del pedido en función vaya avanzando su elaboración o bien marcarlo como entregado cuando llegue al cliente o bien eliminarlo si es necesario.

4.2.2. Clientes

Este usuario corresponde a todo aquel usuario que quiera realizar un pedido en el establecimiento que tenga instalado el sistema.

El sistema guardará la información de los clientes para guardar el historial de los pedidos que se han realizado meramente con motivos organizativos. El correcto uso y custodia de estos datos queda bajo la responsabilidad del gerente del establecimiento.

4.2.2.1. *Ejemplos de uso*

El primer mensaje que recibirá si no escribe ninguno de los comandos preestablecidos será de presentación del establecimiento, indicándole que mensajes puede introducir y que recibirá como respuesta a cada uno de ellos.

Al principio, el sistema le pedirá al usuario que introduzca si quiere que el pedido sea para recoger o a domicilio, en caso de que el usuario escoja “Domicilio”, el sistema pasara a esperar que el usuario escriba la dirección a la que desea que se envié su pedido y este confirme la dirección introducida.

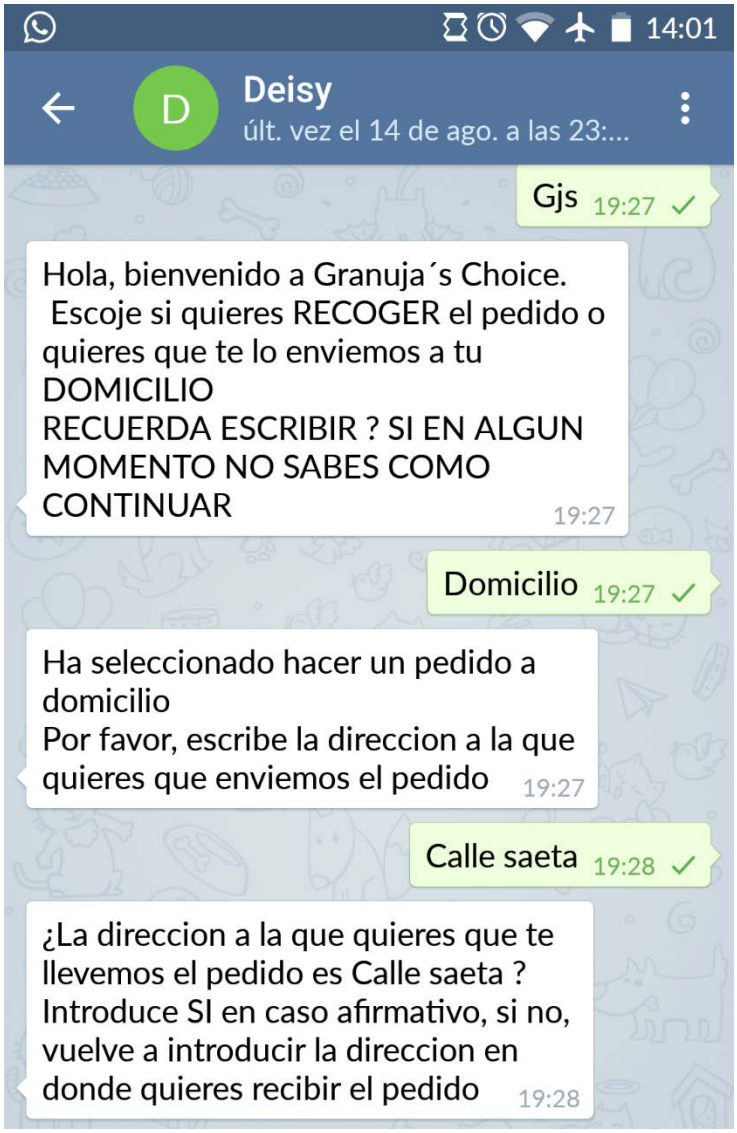


Figura 30: Pantalla de comienzo de un pedido a domicilio

Una vez que el usuario confirma que la dirección que con la que le contesta el sistema es la suya, comienza la creación del pedido. En la imagen siguiente, el usuario confirma que la

dirección que ha introducido es la correcta y pide directamente uno de los platos que existen en la carta, ya que conoce previamente los platos que sirve el restaurante.

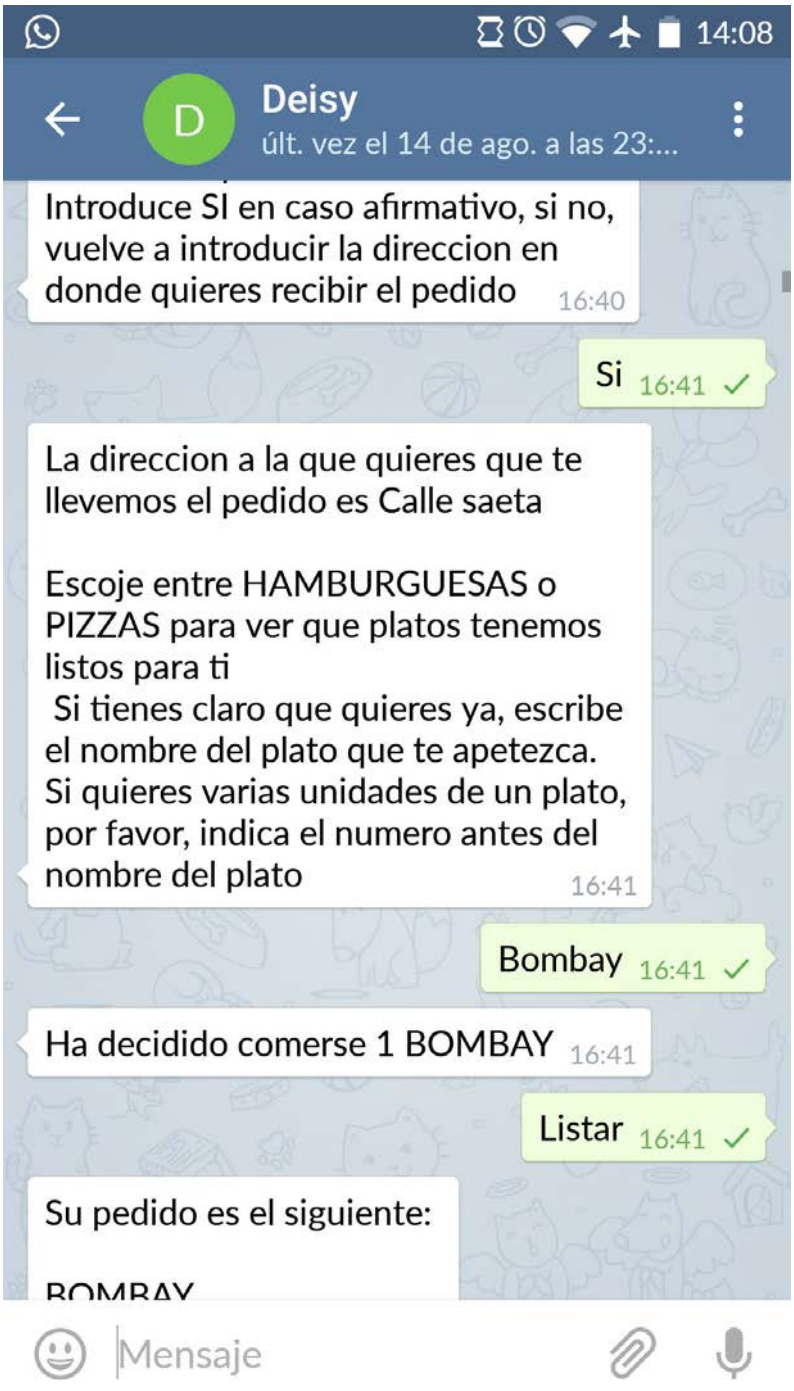


Figura 31: Pantalla incluyendo un plato al pedido

Cuando ha comenzado el pedido, el usuario puede consultar los productos que oferta el restaurante mediante los mensajes “Hamburguesas” o “Pizzas”. Los posibles mensajes que podría aceptar el sistema dependerán de los tipos de productos que estén almacenados en la base de datos.

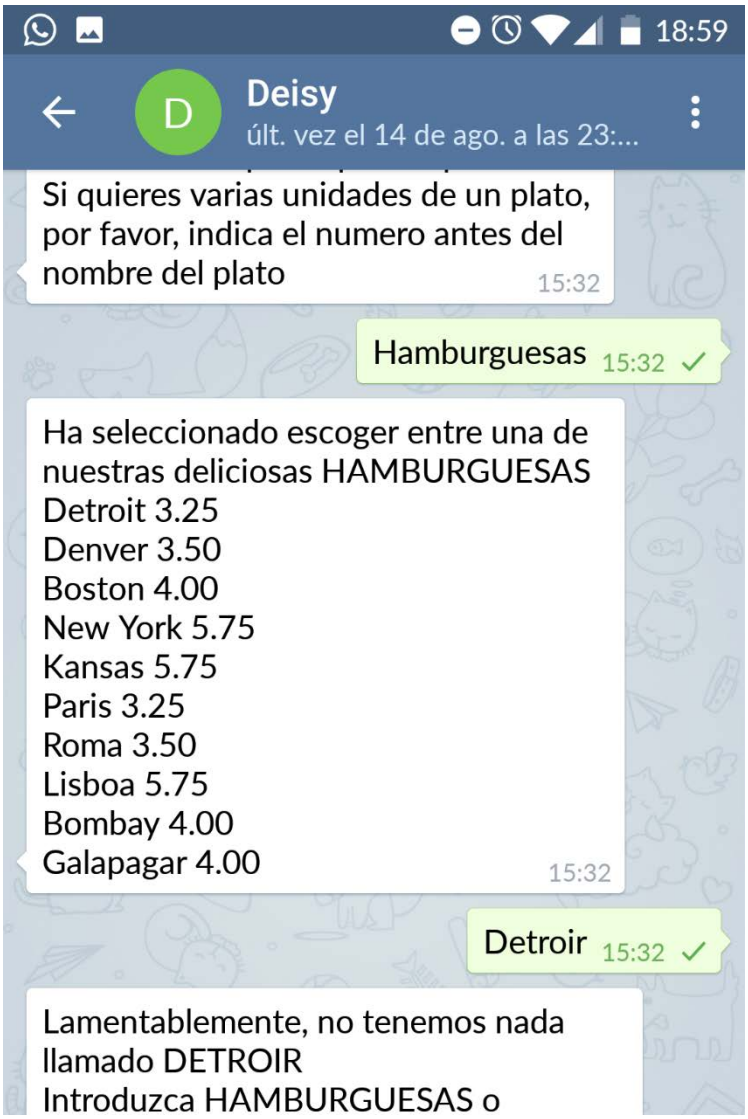


Figura 32: Ejemplo de listado de platos

En caso de que introduzca un mensaje distinto a los indicados, el sistema le avisará del problema y le volverá a escribir que mensajes puede introducir y qué efecto tendrá cada uno de ellos.

Si el usuario consulta su lista de platos y desea eliminar alguno de ellos, tan solo debe escribir “borrar” seguido del nombre del plato que quiere eliminar. El sistema le contestará confirmando que el plato se ha eliminado.

No se ha añadido confirmación al borrado de platos para conseguir que el proceso sea lo más ágil posible. Es difícil que un usuario escriba por error borrar seguido de un plato cuando no desea eliminarlo

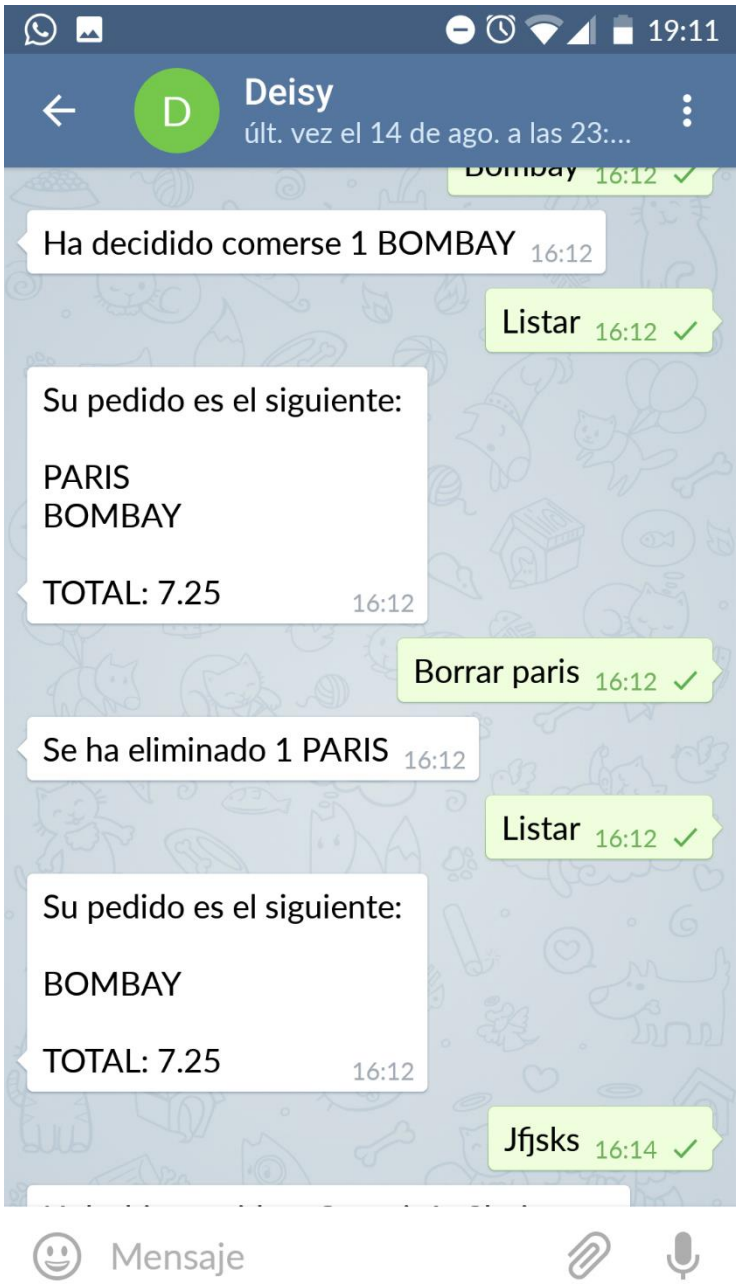


Figura 33: Ejemplo de eliminación de un plato del pedido

Una vez que el cliente haya añadido todos los platos que considere, el usuario solo tendrá que escribir “fin” y confirmar que desea terminar el pedido.

Se le pedirá que confirme la finalización del pedido ya que después de este momento, el pedido se da como realizado a instancias de que aparezca en la interfaz de los cocineros para que estos empiecen su elaboración.

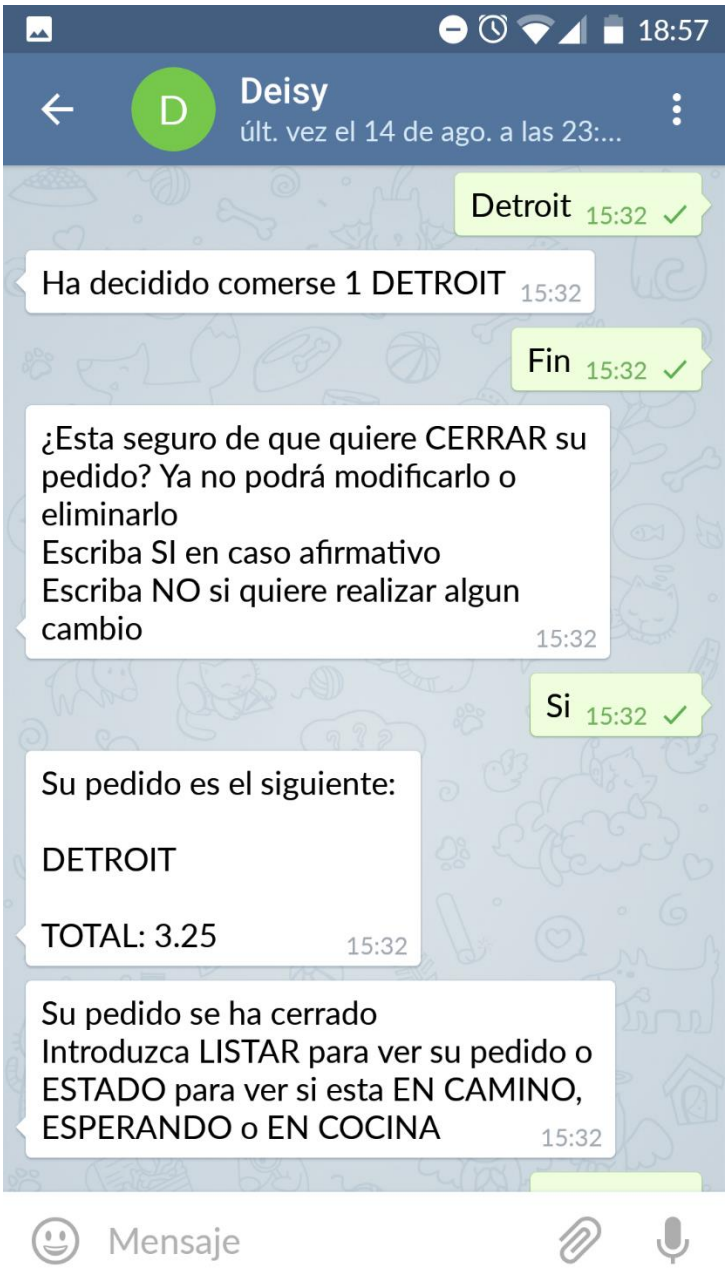


Figura 34: Finalización del pedido

Una vez dado por finalizado el pedido, el sistema quedará a la espera del mensaje pidiendo información del estado del pedido.

Hasta que el pedido no quede marcado como entregado, no se podrán realizar más pedidos desde ese número de teléfono. Esto se debe al flujo del programa, no se da por cerrado completamente un pedido hasta que este esté marcado como entregado, momento en el que la información se copia a la tabla pedidos_fin y se elimina de la tabla pedidos.

5. Planificación y presupuesto:

En este apartado se analizará el presupuesto del proyecto, desde el punto de vista del hardware, el importe del software utilizado, así como los costes asociados al desarrollo del software que hace funcionar el sistema.

Al final del apartado se incluye un diagrama de Gantt para ilustrar la planificación que se ha seguido.

5.1. Presupuesto

5.1.1. Hardware

5.1.1.1. Hardware del sistema:

	Ítem				
	Raspberry Pi	Carcasa con ventilador	Tarjeta SD 64Gb	Cargador	Total
Coste	40,30 €	7 €	29 €	5 €	81,30 €

Tabla 78: Presupuesto del hardware del sistema

Hemos considerado que en un principio no es necesario tener una pantalla conectada al sistema de forma continua, por lo que no está incluido en los costes.

La pantalla solo sería necesaria para la instalación y podría llevarla el encargado de configurar el equipo, el resto del tiempo el equipo podrá funcionar sin tener una pantalla conectada, si el propietario del establecimiento así lo desea, se le suministrará con el equipo una pantalla que se adapte a sus necesidades añadiendo el coste correspondiente.

5.1.1.2. Equipo para desarrollo:

Sistema Operativo	• Ubuntu Linux 16.04
CPU	• Intel Core i7-6700HQ
RAM	• 8.00 GB Dual-Channel
Gráficos	• Nvidia GeForce GTX 960M 2GBs • Intel HD Graphics 530
Almacenamiento	• 1TB HDD

Tabla 79: Especificaciones del equipo utilizado para el desarrollo

El equipo es un ASUS GL552V, con un coste de 850 € en 2016. Se aplica un periodo de amortización de 48 meses. El equipo ha sido utilizado en el desarrollo durante 9 meses, por lo tanto, el coste imputable es de **159,38 Euros**.

5.1.1.3. Teléfono móvil:

El móvil que se ha utilizado principalmente para realizar pruebas durante el desarrollo es un OnePlusOne.

Sistema Operativo	• Android Marshmallow 6.0.1
CPU	• Quad-core a 2,5 GHz
Chipset	• Qualcomm Snapdragon 801
RAM	• 3.00 GB
Gráficos	• Qualcomm Adreno 330
Almacenamiento	• 64 GB

Tabla 80: Especificaciones del móvil utilizado para las pruebas

El coste del equipo fue de 330 Euros, calculamos la amortización a 24 meses. Al igual que el equipo informático, ha sido utilizado durante 9 meses, así que le imputamos un coste de **123,75 Euros**.

5.1.1.4. Coste total del hardware:

Componente	Precio
Hardware del sistema	81,3
ASUS GL552V	159,38
OnePlusOne	123,75
TOTAL	364,43

Tabla 81: Coste total del hardware utilizado en el desarrollo del sistema

5.1.2. Software utilizado:

Los costes asociados al software utilizado consisten en las licencias necesarias para poder utilizarlo.

La gran mayoría del software que hemos utilizado era de libre uso.

La única licencia que ha sido necesario adquirir es la de Office 365 Personal, esta licencia tiene un coste de 7 Euros por mes, su uso ha sido de 9 meses, por lo tanto, el coste total de la licencia es igual a 63 Euros.

5.1.3. Software desarrollado:

En este apartado se detallan los costes del equipo de desarrollo.

El sistema ha sido desarrollado por un único programador, se han imputado los costes de su trabajo por horas.

En función de la tarea que desempeñase, los costes asociados a ella varían, por lo tanto, hemos calculado el coste total de cada una de ellas en función a las horas invertidas en cada tarea.

- Análisis del sistema: 35 Horas
- Diseño del sistema: 90 Horas
- Desarrollo del sistema: 200 Horas
- Pruebas: 80 Horas
- Documentación: 60 Horas

Tarea	Horas	Coste por hora	Total
Análisis del sistema	35	20	700
Diseño del sistema	90	17	1530
Desarrollo del sistema	200	22	4400
Pruebas	80	15	1200
Documentación	60	11	660
Total	**465**		**8490**

Tabla 82: Coste de implementación en función de la tarea

5.1.4. Coste total del sistema:

Teniendo en cuenta los apartados anteriores, el coste total sería el coste agregado de los distintos componentes del sistema.

Concepto	COSTE	Imprevistos (10%)	Coste + Imprevistos
Hardware	364,43	36,44	400,873
Software utilizado	63	0	63
Software desarrollado	8490	849	9339
TOTAL	**8917,43**	**885,44**	**9802.87**

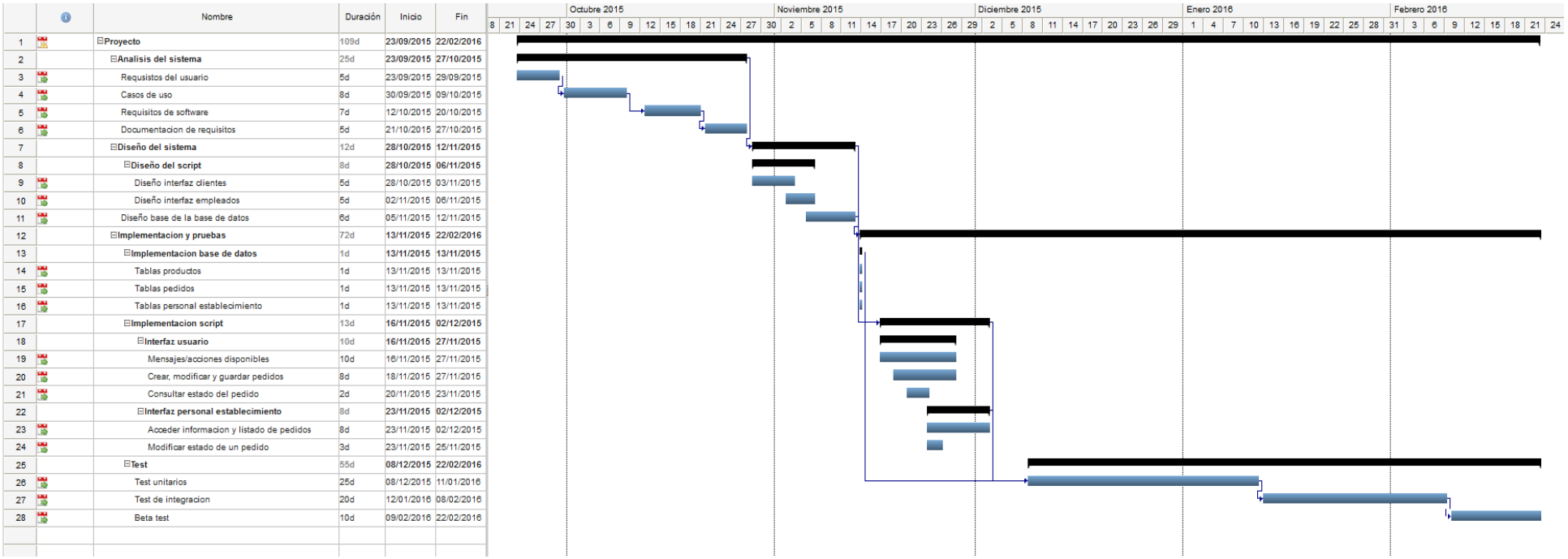
Tabla 83: Presupuesto total del desarrollo

El presupuesto total es de NUEVE MIL OCHOCIENTOS DOS CON OCHENTA Y SIETE Euros.

5.2. Planificación:

Se ha utilizado un diagrama de Gantt con las diferentes fases del desarrollo para ilustrar de una forma gráfica la planificación.

Mediante este diagrama se pueden ver la duración de cada fase y las dependencias que existen entre cada una de ellas.



	Nombre	Duración	Inicio	Fin
1	**Proyecto**	109d	**23/09/2015**	**22/02/2016**
2	**Analisis del sistema**	25d	**23/09/2015**	**27/10/2015**
3	Requisitos del usuario	5d	23/09/2015	29/09/2015
4	Casos de uso	8d	30/09/2015	09/10/2015
5	Requisitos de software	7d	12/10/2015	20/10/2015
6	Documentacion de requisitos	5d	21/10/2015	27/10/2015
7	**Diseño del sistema**	12d	**28/10/2015**	**12/11/2015**
8	**Diseño del script**	8d	**28/10/2015**	**06/11/2015**
9	Diseño interfaz clientes	5d	28/10/2015	03/11/2015
10	Diseño interfaz empleados	5d	02/11/2015	06/11/2015
11	Diseño base de la base de datos	6d	05/11/2015	12/11/2015
12	**Implementacion y pruebas**	72d	**13/11/2015**	**22/02/2016**
13	**Implementacion base de datos**	1d	**13/11/2015**	**13/11/2015**
14	Tablas productos	1d	13/11/2015	13/11/2015
15	Tablas pedidos	1d	13/11/2015	13/11/2015
16	Tablas personal establecimiento	1d	13/11/2015	13/11/2015
17	**Implementacion script**	13d	**16/11/2015**	**02/12/2015**
18	**Interfaz usuario**	10d	**16/11/2015**	**27/11/2015**
19	Mensajes/acciones disponibles	10d	16/11/2015	27/11/2015
20	Crear, modificar y guardar pedidos	8d	18/11/2015	27/11/2015
21	Consultar estado del pedido	2d	20/11/2015	23/11/2015
22	**Interfaz personal establecimiento**	8d	**23/11/2015**	**02/12/2015**
23	Acceder informacion y listado de pedidos	8d	23/11/2015	02/12/2015
24	Modificar estado de un pedido	3d	23/11/2015	25/11/2015
25	**Test**	55d	**08/12/2015**	**22/02/2016**
26	Test unitarios	25d	08/12/2015	11/01/2016
27	Test de integracion	20d	12/01/2016	08/02/2016
28	Beta test	10d	09/02/2016	22/02/2016

Figura 35: Diagrama de Gantt

6. Futuras mejoras:

Este apartado contiene aquellas posibles modificaciones o funcionalidades que han ido surgiendo durante el desarrollo y el diseño del actual sistema y que podrían añadirse en un futuro.

Algunas de ellas consisten en añadir pequeñas funcionalidades al sistema y otras implicarían cambios más profundos en el diseño y estructura del este. Todas ellas se han planteado como un ideario y requeriría de una investigación más exhaustiva para conocer lo factible que sería su implementación real.

6.1. Enviar geolocalización:

Esta modificación está inspirada en uno de los servicios que ofrece una de las aplicaciones móviles explicadas en el apartado **Estado del arte**.

Deliveroo ofrece a sus clientes la posibilidad de consultar en todo momento la posición del repartidor encargado de llevarle su pedido.

Nosotros podríamos incluir esta misma funcionalidad en el sistema. La información de la posición del repartidor del establecimiento podría ser compartida por el dispositivo móvil de este siempre que tuviera GPS.

Una opción sería que el repartidor publicará en donde se encuentra en todo momento en una página web mediante APIs de geolocalización, aunque esto obligaría a hacer pública esta información o crear un sistema de usuarios para que cada cliente pueda acceder solo a la información de su repartidor. Este método parece un tanto complicado y farragoso como para poder ser implementado de una forma sencilla.

Otra solución sería compartir las coordenadas del repartidor periódicamente para que el cliente pudiera consultar en google a que posición del mapa corresponden, sin embargo, esta solución parece poco cómoda para el cliente.

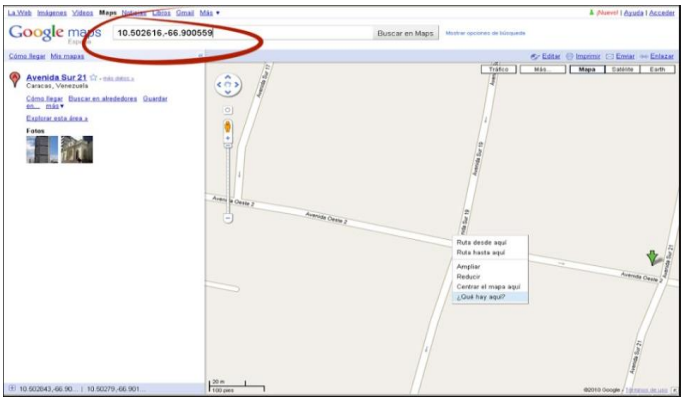


Figura 36: Ejemplo de búsqueda de ubicación por coordenadas en Google Maps

La solución que podríamos realizar, sin añadir el uso de otras aplicaciones para acceder a nuestros servicios sería combinar el uso de una aplicación desarrollada por nosotros para que el dispositivo del repartidor envíe a la base de datos periódicamente su ubicación en coordenadas y que el script se encargara de acceder a esa información, traducirla a una calle y enviársela al cliente cuando este lo requiera.

Si el cliente tiene instalada alguna aplicación como *GoogleMaps*, es posible que pudiera enviarse un mensaje de tal forma que al pulsar sobre él se lance la aplicación de mapas y esta le muestre el punto que corresponde a las coordenadas enviadas al sistema por el repartidor.

Según la documentación actual del cliente de *Telegram* que estamos usando, no soporta compartir la ubicación de los interlocutores por lo que no sería posible incluir un mensaje del estilo de *WhatsApp* o la versión móvil de *Telegram* en donde se envía un mensaje con un mapa cuando compartimos nuestra ubicación.

Durante el desarrollo de la aplicación hemos encontrado que la herramienta tiene más funcionalidades de las descritas en la documentación, por lo que no descartamos que con algo de investigación se pudiera conseguir que el script mandara un mensaje similar al mencionado previamente.

Otra solución sería utilizar aplicaciones de terceros para tratar de integrar sus soluciones con nuestro sistema

Existen varias aplicaciones en el mercado para poder compartir en tiempo real la ubicación GPS de un dispositivo, sin embargo, este método obliga al restaurante y al cliente cambiar de entorno para permitir que el repartidor informe de su posición en todo momento.

Google+ y Facebook ofrecen la posibilidad de compartir la ubicación de un usuario de la red social, sin embargo, debemos tener agregado como contacto a aquellos usuarios con los que queremos compartir la ubicación.

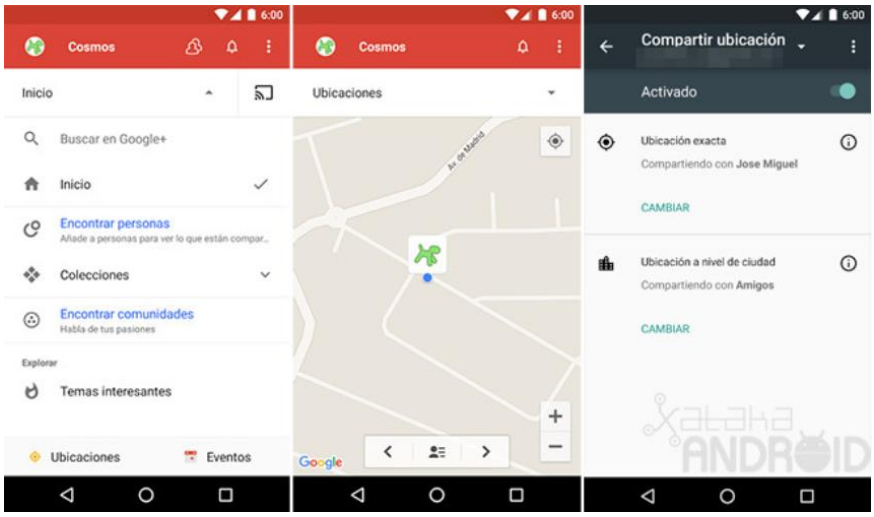


Figura 37: Ejemplo de ubicación compartida en Android

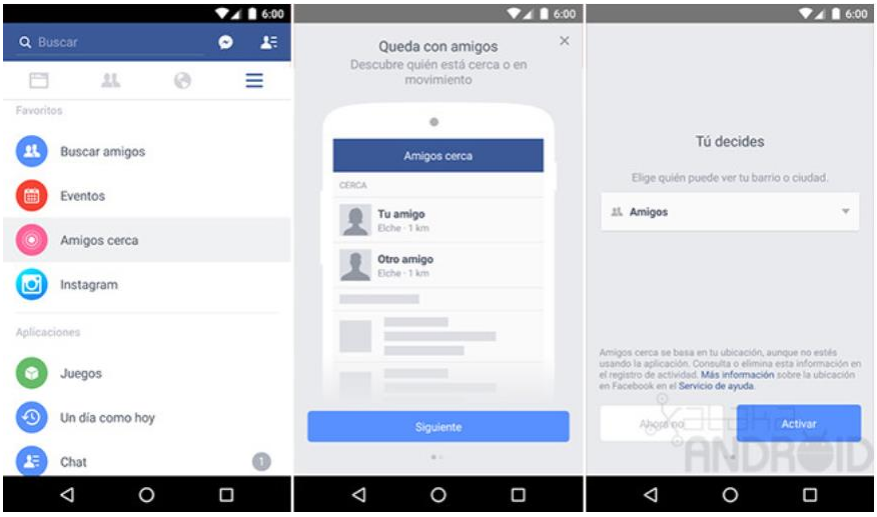


Figura 38: Ejemplo de ubicación compartida mediante la aplicación de Facebook

Waze es otra aplicación que ofrece la posibilidad de compartir la ubicación, entre otras muchas funcionalidades, está orientada a ser una red social de conductores, permitiendo a sus usuarios notificar accidentes o zonas de atascos. A pesar de ser una aplicación muy potente, se aleja del objetivo que buscamos, ya que compartir la ubicación solo es una funcionalidad secundaria de esta, además, es necesario que los contactos con los que quieras compartir tu ubicación tengan la aplicación instalada en el dispositivo. Esto representa una desventaja

frente a las dos anteriores alternativas, ya que esta aplicación, a diferencia de las otras, no suele venir instalada por defecto en los terminales móviles.

Figura 39: Pantallas de Waze

Comparando con el resto de alternativas analizadas, recomendaríamos al establecimiento que utilizara *Glympse*.

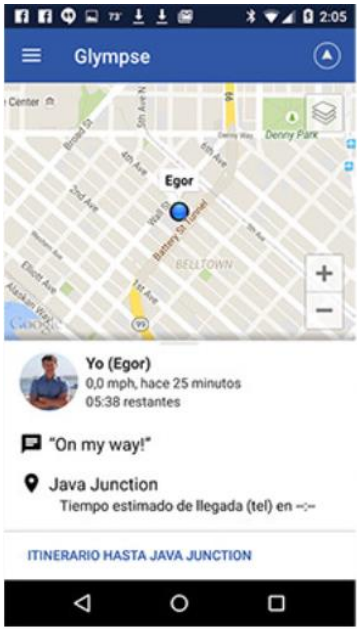


Figura 40: Pantalla de Glympse

Esta aplicación permite compartir la ubicación durante un tiempo determinado, de 5 minutos a 4 horas.

Podemos compartir la ubicación con usuarios que ya tengan la aplicación instalada, en cuyo caso podrán ver el movimiento del repartidor mediante la aplicación.

Si el contacto no tiene la aplicación instalada, podemos enviarle por SMS un link a la página web de la aplicación en donde aparecerá un mapa y un punto que indica la posición actualizada del usuario que ha compartido su ubicación.

Este link podría enviarse mediante la interfaz del sistema desarrollado para permitir que el propietario del restaurante ahorrase los costes asociados al envío de SMS. Para realizar el envío del link, el repartidor podría seleccionar la opción de compartir antes de ponerse en marcha, copiar el link que le indica la aplicación y enviárselo al sistema mediante un mensaje de *Telegram*, en cuanto esto se produjera, el sistema mandaría automáticamente un mensaje informando al usuario que el repartidor se ha puesto en camino y que puede consultar donde se encuentra mediante el link que se le facilita a continuación.

La vista que tendría el cliente del restaurante si accediese a la ubicación del repartidor mediante la web sería algo similar a la siguiente imagen.

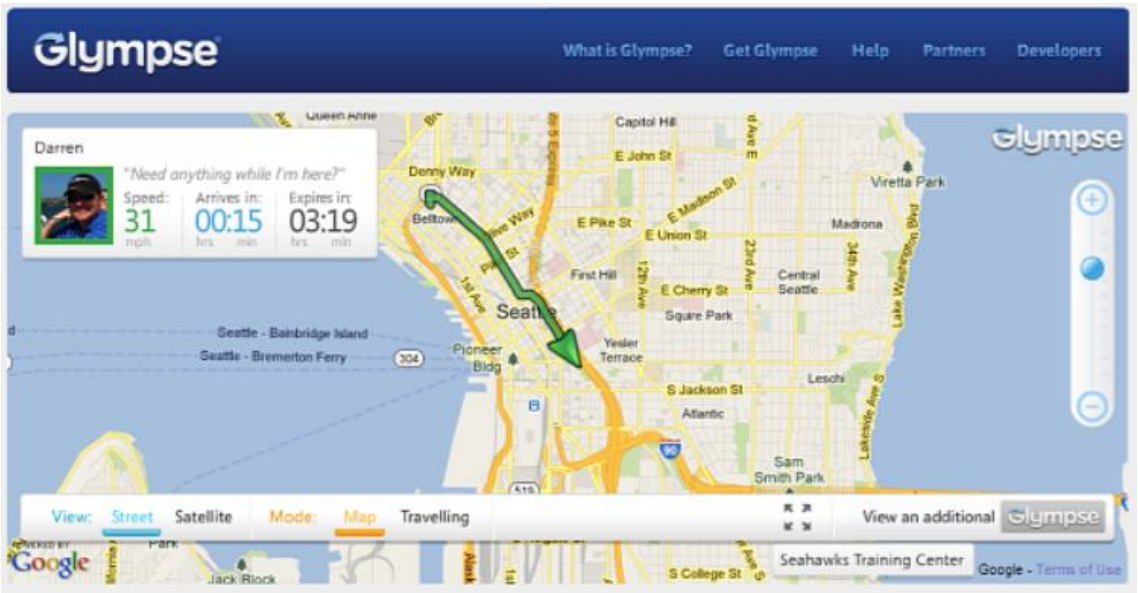


Figura 41: Mapa con trayecto de un usuario de Glympse

En la imagen, el contacto que comparte la ubicación ya es conocido, por lo que aparece con su foto, además, se ha configurado la aplicación para informar al receptor de cuál es el destino y esta ha calculado el tiempo estimado de llegada.

Gracias al uso de esta aplicación, podríamos añadir la funcionalidad de informar al cliente del tiempo estimado en el que su pedido llegará.

Glympse se anuncia como una aplicación gratuita y según hemos visto, su uso es gratuito para fines empresariales también, sin embargo, existe un apartado dentro de la web, llamado **"Glympse for Business"**, en ella anuncian la existencia de APIs y un entorno de desarrollo para facilitar la integración.

6.2. Enviar automáticamente cambio de estado de pedido:

Esta mejora consiste en añadir en el código del script código que se encargue de informar al receptor de un pedido concreto el momento en el que alguno de los encargados del restaurante ha modificado el estado de su pedido, cuando cocina empiece a prepararlo y lo marque como "En cocina", el cliente recibirá un mensaje automáticamente avisándole de este cambio, y así con todos los estados por los que pase su pedido.

Para poder enviar estos mensajes automáticos, sería necesario almacenar el nombre de usuario aparte de su número de teléfono, por lo que habría que incluir un atributo más en la tabla de cliente en donde se guardaría este valor.

Una vez creada la tabla y almacenado el nombre de usuario, el sistema accedería a este y podría escribir al cliente automáticamente cuando su pedido recibiera algún cambio.

Esta modificación se podría ampliar para que el cliente recibiera notificaciones cada vez que uno de los platos de su pedido se haya completado, y luego un mensaje final cuando el ultimo plato se haya preparado y todo el pedido esté listo para ser repartido. Para ello, habría que incluir un nuevo atributo en la tabla "Cuenta_producto" para representar si el producto se encuentra preparado o no.

6.3. Reconocedor de textos "quizás quiso decir":

Con el fin de conseguir que los usuarios puedan recibir en todo momento retroalimentación del sistema y mantener su uso en consonancia con las buenas prácticas de interfaces de usuario ante cualquier mensaje no reconocido, el sistema contestará con los mensajes que pueden realizar cambios en su estado e información de los cambios que cada uno de esos mensajes generarán.
Esta forma de interacción, realizada de la forma correcta, asegura que el usuario pueda avanzar a través de todo el proceso sin perderse ni sentirse frustrado. Sin embargo, puede causar problemas ante situaciones como escribir mal el nombre de un plato por una letra o

confundir un comando por pulsar una tecla cercana a la que quería el empleado del restaurante.

Para corregir este problema, sería interesante añadir una funcionalidad encargada de "adivinar" que es lo que quería decir el usuario. De la misma forma que el buscador Google sugiere búsquedas cuando no termina de reconocer lo que le hemos escrito, podríamos agilizar la interacción de los usuarios si el sistema contestara con una sugerencia de texto en vez de requerir que se le escriba de una forma totalmente correcta el mensaje que esperaba recibir.

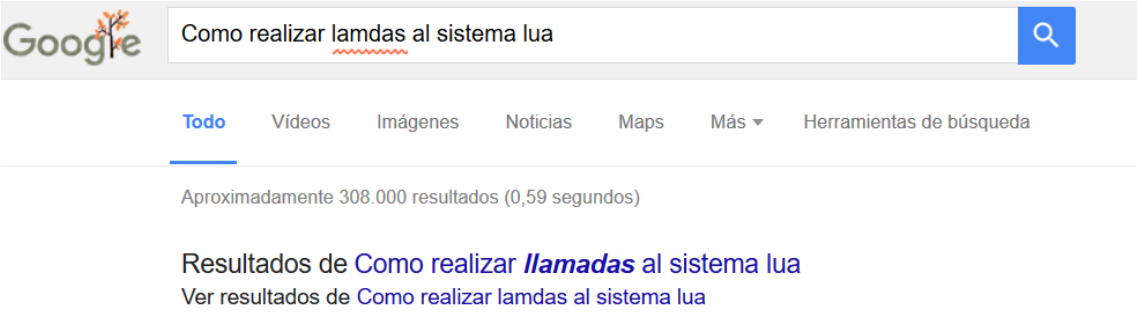


Figura 42: Pantalla de sugerencia de búsquedas de Google

Cuando el sistema realizase una sugerencia, el usuario solo tendría que confirmar si el mensaje que ha supuesto el sistema es lo que él quería poner.

Esta funcionalidad cobraría especial importancia en la interfaz de los clientes, ya que los mensajes que deberán introducir son más largos, lo que aumenta las probabilidades de cometer un error de escritura y un mayor tiempo para poner el mensaje correcto.

6.4. Modificación de platos y comentarios:

Cuando se realiza un pedido por teléfono o mediante aplicaciones o internet, es bastante usual que los restaurantes hayan contemplado que los clientes quieran introducir modificaciones en los platos que han escogido. Estas modificaciones pueden ser, por ejemplo, que se ponga mucha salsa, pedir la hamburguesa sin pepinillos o sin cebolla.

Para que esta funcionalidad estuviese presente en el sistema, podríamos incluir un campo de comentarios en donde el cliente pudiera incluir texto con todas las directrices que considerara oportuno.

Una posible solución implicaría añadir un campo en la tabla "Cuenta_producto" para almacenar los comentarios plato por plato. Después de añadir cada producto al pedido, el cliente recibiría un mensaje preguntándole si quiere añadir algún comentario al producto que

acaba de añadir y después de confirmar que quiere hacerlo, de forma similar a la solución anterior, el cliente podría escribir texto indicando que pequeñas modificaciones debería tener el plato.

Esta solución parece que resta agilidad al proceso de elaborar el pedido, a pesar de que permite tener ordenado por plato los comentarios, es poco probable que cada uno de los pedidos tenga que tener alguna directriz especial. Esta solución podría ser optima si existe una gran variedad en cada uno de los productos, es decir, cada uno de los platos que ofrece el establecimiento tiene muchas posibles variaciones.

Otra solución para este problema sería añadir un campo en la tabla “Cuenta” de nombre “comentario” en donde se guardaría este texto.

Esta solución resulta interesante en el estilo de carta que hemos utilizado para ilustrar el funcionamiento y el diseño en este trabajo. Existen platos diferenciados entre si y fácilmente reconocibles unos de otros, las modificaciones serian casos concretos de cada uno de los platos, ya sea pedir la carne poco o muy hecha, más o menos salsa u omitir alguno de los ingredientes del plato.

Una vez que en el establecimiento reciben la notificación del pedido o alguno de los empleados decide consultar los detalles de este, el sistema les mandaría, aparte de toda la información que se envía hasta ahora, el comentario que ha añadido cada cliente a su pedido. De esta forma, el personal conoce desde el primer momento si existen directrices especiales y en qué consisten estas.

6.5. Pasarela de pago:

Basándonos en el modelo de negocio de algunos de nuestros competidores, se podría integrar el sistema con una pasarela de pago.

El cliente podría introducir las credenciales de sistemas de pago como cuentas bancarias o PayPal, nosotros nos encargaríamos de gestionar y validad el pago a cambio de una pequeña comisión. Esto permitiría al cliente olvidarse del pago una vez que ha completado el pedido, cuando cerrase su móvil, solo tendría que preocuparse de abrir la puerta cuando el repartidor llegara a su casa.

Para incluir esta funcionalidad, tendríamos que contratar los servicios de alguna empresa especializada en la gestión de datos tan sensibles, llegar a acuerdos con las entidades bancarias y asegurar nuestra base de datos de una forma mucho más robusta que la necesaria para almacenar datos como la dirección o el teléfono.

Este sistema podría ser una nueva fuente de financiación para la empresa aparte de la generada por los sistemas, esto permitiría un mayor desarrollo de la empresa y aumento de las capacidades de esta, consiguiendo más capital para poder adquirir más equipos o ampliar el equipo del desarrollo.

Aparte de las entidades bancarias PayPal puede ser integrado en páginas web y si no se dispone de pasarela de pago para procesar tarjetas, pueden encargarse de gestionar las operaciones con tarjetas también.

Figura 43: Imagen de PayPal

El proceso puede realizarse a través del teléfono móvil, lo que facilitaría su uso junto a nuestra aplicación.

Un ejemplo de uso podría ser ofrecer al cliente las diferentes formas de pago, en el caso de que escoja la opción de PayPal, se le enviaría un link con la referencia a la cuenta del establecimiento y el cliente, ya en la plataforma de PayPal introduciría sus credenciales o su cuenta bancaria desde la que quiere realizar el pago.

PayPal ofrece este servicio a empresas y cobra por ello

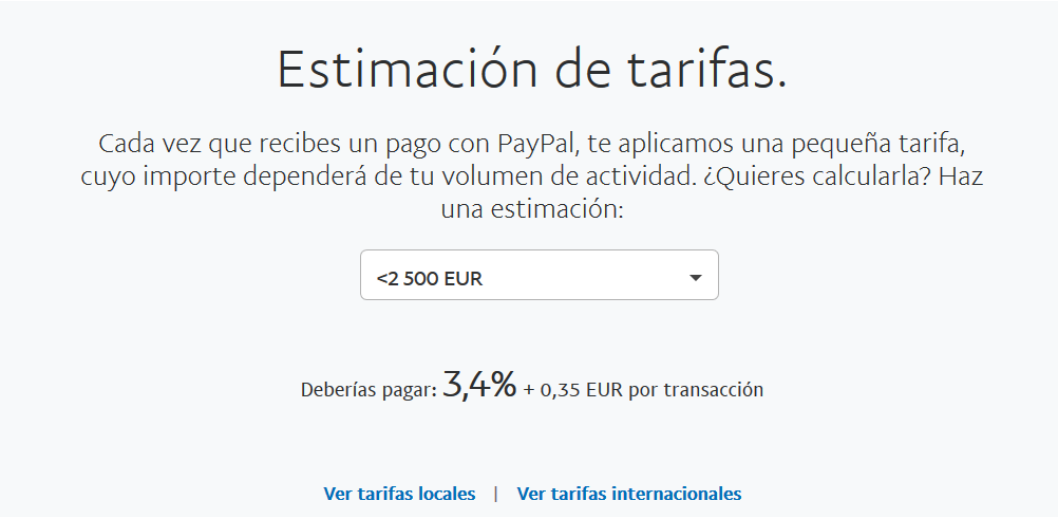


Figura 44: Estimación de tarifa cobrada por PayPal

Las tarifas van siendo menores cuanto mayor es el volumen de negocio de la empresa, lo que quizás obligaría a poner restricciones a los clientes de pequeños establecimientos en el importe mínimo que sería necesario para utilizar PayPal.

Sería necesario realizar un análisis económico para averiguar en hasta qué punto encargarse de la gestión de estas operaciones podría resultar rentable. Sin embargo, es un punto interesante, ya sea para obtener nuevas vías de financiación o simplemente para aprovechar la potencia de PayPal para hacer nuestro producto más atractivo y ofrecer al cliente nuevas formas de realizar los cobros.

6.6. Integración de TPV:

En consonancia con el punto anterior, sería interesante integrar todo lo posible nuestro sistema en la infraestructura que el establecimiento tenga montada previamente.

Actualmente, la mayoría de los establecimientos de comida con los que nos encontramos cuentan con dispositivos para realizar cobros mediante tarjeta de crédito, algunos incluso tienen dispositivos móviles para que sus repartidores puedan cobrar con ese método cuando salen del establecimiento a entregar un pedido.

Aparte del dispositivo encargado del cobro mediante tarjeta de crédito, los establecimientos pueden tener un equipo informático para gestionar la caja con dinero físico, las existencias y llevar la anotación de los pedidos que se han ido realizando durante la jornada. Al conjunto de dispositivos que forman este sistema se le denomina TPV.

Para interactuar con estos equipos existe un estándar llamado Odoo. Este software consiste en un gestor empresarial, que se encarga de todos los aspectos que podrían influir en un negocio,

desde zona de clientes en la web del establecimiento hasta la contabilidad o la gestión del inventario.

Odoo es software de código abierto separado en distintos módulos en según la funcionalidad que estos ofrezcan, estos módulos, aunque independientes entre sí, están completamente integrados entre ellos. Uno de los módulos de este software es el que se encarga del uso de los TPV y permite que se realicen las gestiones desde distintos dispositivos. Para su funcionamiento solo requiere de un navegador web y conexión a internet, aunque en el caso de que la conexión se cayera, automáticamente actualizaría los datos en cuanto volviera a tener conexión.

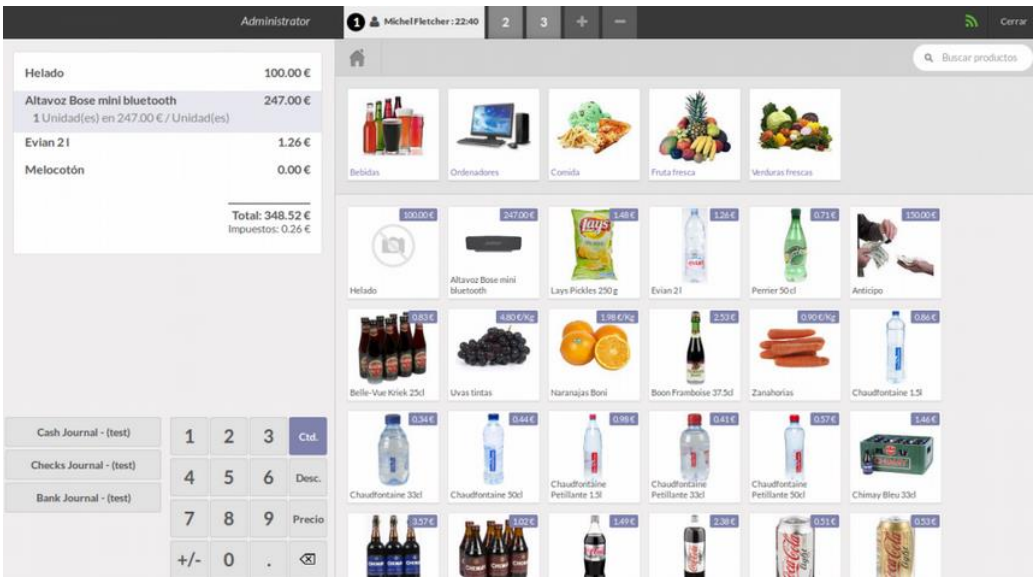


Figura 45: Pantalla de módulo TPV

El módulo TPV se podría encargar de obtener de la base de datos utilizada por nuestro sistema los pedidos que se han realizado y utilizar esa información con el resto de módulos, de esta forma, de manera automáticamente los pedidos quedarían anotados para la contabilidad y la gestión de inventario de la misma forma que se realiza este proceso cuando los empleados utilizan el TPV del establecimiento.

Que el código sea abierto nos permite analizarlo y encontrar una manera de poder combinar ambos sistemas, este proceso, una vez realizado conseguiría colocar nuestro gestor de pedidos para restaurantes en una buena posición para conseguir entrar en nuevos establecimientos y mejorar el servicio ofrecido a los clientes con los que ya tuviéramos un acuerdo comercial.

6.7. Añadir información nutricional:

Cada vez más los consumidores se sienten preocupados por el valor nutricional de los productos que consumen. Un ejemplo de esta tendencia es la iniciativa de cadenas de comida rápida como McDonald`s, que ha incluido información nutricional de los productos que sirve.

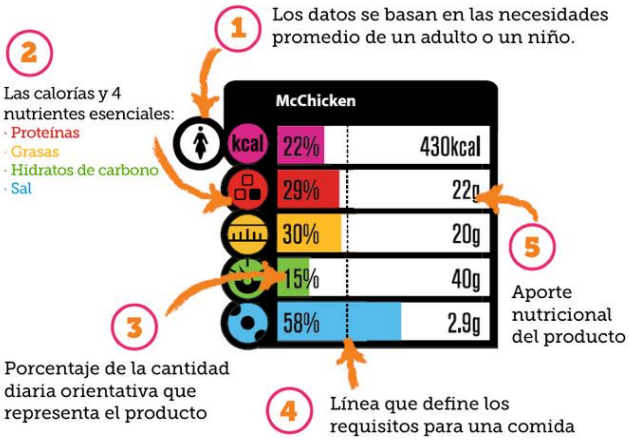


Figura 46: Información nutricional de McChicken

Puede ser un factor interesante a la hora de atraer clientes para el establecimiento que estos pueda consultar de forma rápida y cómoda información como las calorías que tienen los platos que se sirven en el restaurante.

Para conseguir esta funcionalidad, podríamos guardar información de las calorías de cada ingrediente utilizado, sin embargo, esto nos obligaría a realizar cálculos internamente teniendo en cuenta calorías por gramo de cada ingrediente y gramos utilizados de cada ingrediente en los platos. El trabajo y la sobrecarga al sistema que esta solución conlleva no compensa con los resultados obtenidos.

Una solución más sencilla e igual de efectiva sería incluir un campo en la tabla de la base de datos “Productos” con el valor calórico de ese plato, habiendo realizado previamente el cálculo, sin necesidad de almacenar información de cada uno de los ingredientes.

Esta solución es más compacta, además, podría ser extensible a añadir los valores de vitaminas, minerales, etc. Tan solo sería necesario añadir nuevos campos a la tabla e incluir código para sacar la información de esos campos cuando el cliente lo requiriese.

Para que el cliente pudiera acceder a esos datos de una forma cómoda y rápida, pensamos que el mejor momento para mostrársela seria junto a la carta de los productos, en el mismo mensaje en donde se muestra el nombre de los productos y el importe de estos se añadiría el valor calórico de cada uno de ellos. Consideramos que el valor calórico es el dato que interesa

principalmente a la mayoría de los clientes, por lo tanto, sería interesante mostrárselo sin necesidad de que lo pidan.

En caso de que incluyésemos más información nutricional, se podría ofrecer al cliente la posibilidad de utilizar un comando nuevo, por ejemplo, “info” seguido del nombre del producto que desea consultar para obtener más datos nutricionales de ese plato. En este mismo apartado podríamos incluir una línea informando de las salsas o añadidos con los que se sirve y el valor nutricional de estas, para que el cliente pudiera descontar las calorías en caso de introducir en los comentarios que no desea alguno de los añadidos.

7. Marco regulatorio:

Para cumplir con la legislación vigente de protección de datos, los datos de los clientes de los restaurantes, así como los números de teléfono de los empleados de los establecimientos deben ser registrados en la Agencia Española de protección de datos (Ley orgánica 15/1999 de 13 de diciembre). Nos acogemos a la Directiva de Protección de Datos 95/46, combinada con la Directiva 2002/58/CE de Privacidad y Comunicaciones Electrónicas (European Comission, 2013).

Para evitar el mal uso de los datos personales de los usuarios, el desarrollo del sistema se encargará de:

- Proteger los datos de los usuarios cifrando el medio de almacenamiento en donde se encuentren.
- No utilizar los datos recogidos para otro fin que el anunciado a los usuarios.
- Garantizar el derecho a acceso, rectificación, oposición y cancelación de nuestros servicios.

Debido a que la base de datos se encontrará custodiada por los encargados del restaurante y toda la información será utilizada para su funcionamiento interno, ellos tendrán que encargarse de salvaguardar los datos de los clientes almacenados en el sistema de la misma forma que los datos que tengan almacenados en cualquier otro medio.

Nosotros nos encargaremos de realizar las modificaciones en la base de datos que sean necesarias, así como dotar de la seguridad suficiente al sistema para que no se puedan obtener información de forma fraudulenta, pero la gestión deberá ser llevada por cada restaurante y será total responsabilidad de ellos que se respete la ley, así como de informar a sus clientes de sus derechos con respecto a la información que se almacenará de ellos.

8. Conclusión:

El sistema implementado permite la gestión de los pedidos a domicilio de un restaurante, ha cumplido con todas las funcionalidades planteadas al comenzar el diseño. Es capaz de realizar la interacción con el cliente de una forma natural y ofrecer a los empleados una interfaz cómoda y rápida para acceder a la información que necesitan.

El sistema es funcional y podría instalarse en todo establecimiento interesado después de discutir con el propietario que adaptaciones concretas querría que se hicieran.

La experiencia de utilizar un lenguaje desconocido, así como código obtenido en un repositorio en donde no hay demasiada documentación ha sido algo ardua, sin embargo, ha sido muy enriquecedora.

Ha servido para conocer de primera mano en que consiste un proyecto de desarrollo con una visión completa, lo que ha permitido utilizar de forma completa muchas de las competencias adquiridas durante el grado e integrarlas de forma conjunta, interactuar con una base de datos desde un programa externo y que esta interacción se haga de forma automática mediante el uso de uno de los sistemas operativos estudiados.

Se ha implementado un sistema que puede convertirse en un producto comercial, en la misma línea que algunas de las aplicaciones que han surgido para facilitar la comunicación e interacción entre los restaurantes y sus clientes, esto significa que desde el punto en donde se ha quedado el sistema podría desarrollarse un sistema más completo que pudiera convertirse en un serio competidor dentro de este mercado.

Durante el proceso de desarrollo han aparecido numerosas modificaciones y posibles planteamientos por los que el sistema podría desembocar, en el apartado “Futuras mejoras” se pueden encontrar algunas de ellas.

El hecho de poder interactuar con una terminal de un equipo Linux, de la misma forma que si el usuario se encargará delante del equipo plantea muchas posibilidades, desde obtener capturas de una cámara y que estas se envíen al móvil, realizar cambios en nuestro ordenador tan solo enviando un mensaje desde nuestro móvil, hasta crear una especie de blog en donde los usuarios interactúan entre ellos y con el equipo mediante la aplicación *Telegram.*

En definitiva, el proceso de creación de este sistema me ha permitido conocer más profundamente herramientas útiles para desarrollar proyectos inexistentes en el mercado actualmente y que podrían abrirse hueco en su mercado gracias a la combinación de novedad

y su profunda relación con las redes sociales, un mercado que se encuentra en auge y no tiene visos de disminuir en popularidad.

9. Bibliografía:

20minutos.es - Últimas Noticias. (2016). *Whatsapp vs Telegram: Las virtudes de Telegram contra la popularidad de WhatsApp*. [online] Available at: http://www.20minutos.es/noticia/2762095/0/whatsapp-telegram-comparativa-mensajeria-instantanea/ [Accessed 7 Jun. 2016].

ABC.es. (2016). *La Nevera Roja cierra una ronda de financiación de 2 millones de euros y consolida su liderazgo en España*. [online] Available at: http://www.abc.es/economia/20130927/abci-nevera-roja-financiacion-201309261316.html [Accessed 17 Feb. 2016].

ADSLZone. (2016). *Fallos de seguridad en el protocolo SS7 hacen que el cifrado de WhatsApp no valga de nada*. [online] Available at: http://www.adslzone.net/2016/05/10/fallos-seguridad-protocolo-ss7-hacen-cifrado-whatsapp-no-valga-nada/ [Accessed 17 May 2016].

Amazon.es. (2016). *Tarjeta SD 64GB Samsung*. [online] Available at: https://www.amazon.es/Samsung-MB-MC64DA-AMZ-Tarjeta-adaptador/dp/B00WIMBZHK/ref=lp_934128031_1_2?s=electronics-accessories&ie=UTF8&qid=1474731655&sr=1-2 [Accessed 21 Sep. 2016].

Applicantes. (2016). *Aplicación La Nevera Roja*. [online] Available at: http://applicantes.com/la-nverea-roja-aplicaciones-ios-android/ [Accessed 16 May 2016].

Arume, a. (2013). *HTML5: API de geolocalización (Geolocation API) - Blog - Arume*. [online] Arumeinformatica.es. Available at: http://www.arumeinformatica.es/blog/html5-api-de-geolocalizacion-geolocation-api/ [Accessed 4 Sep. 2016].

Barometro 2015 Just Eat. [online] Available at: https://www.just-eat.es/adomicilio/images/barometro_2015.pdf [Accessed 18 Mar. 2016].

Bytacora Soluciones Informáticas. (2015). *El TPV en Odoo - Bytacora Soluciones Informáticas*. [online] Available at: http://www.bytacora.es/openerp-2/el-tpv-en-odoo/ [Accessed 21 Sep. 2016].

DELAVEGA, Cesar. *Desarrollo de aplicación móvil de seguimiento de personas en tiempo real*. Madrid, 2015.

Es.python.org. (2016). *Asociación Python-España*. [online] Available at: http://www.es.python.org/ [Accessed 26 Sep. 2016].

Fernández, S. (2016). *España, territorio smartphone*. [online] Xatakamovil.com. Available at: http://www.xatakamovil.com/movil-y-sociedad/espana-territorio-smartphone [Accessed 13 Feb. 2016].

GitHub. (2016). *vysheng/tg*. [online] Available at: https://github.com/vysheng/tg [Accessed 20 Sep. 2016].

Glympse.com. (2016). *Glympse for Business*. [online] Available at: https://www.glympse.com/business [Accessed 13 Sep. 2016].

Linux-es.org. (2016). *Sobre Linux | El rincón de Linux*. [online] Available at: http://www.linux-es.org/sobre_linux [Accessed 17 May 2016].

Lua.org. (2016). *Lua: about*. [online] Available at: https://www.lua.org/about.html [Accessed 25 Sep. 2016].

McDonalds, (2016). [online] Available at: https://www.mcdonalds.es/calidad/informacion-nutricional/nutricion-mcdonald-s [Accessed 21 Sep. 2016].

Odoo - OpenERP - ERP, CRM, MRP, SGA 100% Libre - | Sin Licencias. (1970). *¿Que es OpenERP? ERP 100% Libre*. [online] Available at: http://openerpspain.com/openerp/que-es-openerp/ [Accessed 21 Sep. 2016].

Pastor, J. (2014). *IDC: el crecimiento explosivo de los smartphones tiene fecha de caducidad*. [online] Xatakamovil.com. Available at: http://www.xatakamovil.com/mercado/idc-el-crecimiento-explosivo-de-los-smartphones-tiene-fecha-de-caducidad [Accessed 17 Feb. 2016].

Pastor, J. (2016). *Cómo funciona el cifrado extremo a extremo de Whatsapp y qué implicaciones tiene para la privacidad*. [online] Xataka.com. Available at: http://www.xataka.com/seguridad/como-funciona-el-cifrado-extremo-a-extremo-de-whatsapp-y-que-implicaciones-tiene-para-la-privacidad [Accessed 10 Apr. 2016].

Paypal.com. (2016). *Venda por Internet y acepte pagos seguros a través PayPal*. [online] Available at: https://www.paypal.com/es/webapps/mpp/accept-payments-online [Accessed 16 Sep. 2016].

Puerto, K. (2014). *OnePlus One, análisis*. [online] Xataka.com. Available at: http://www.xataka.com/analisis/oneplus-one-analisis [Accessed 21 Sep. 2016].

Sánchez, A. (2016). *WhatsApp sigue imparable: mil millones de usuarios activos*. [online] Hipertextual. Available at: https://hipertextual.com/2016/02/whatsapp-maus-2016 [Accessed 5 May 2016].

Santos, E. (2016). *Los 13 mejores bots que puedes añadir a Telegram*. [online] Genbeta.com. Available at: http://www.genbeta.com/mensajeria-instantanea/los-13-mejores-bots-que-puedes-anadir-a-telegram [Accessed 19 May 2016].

Solo Codigo Web. (2014). *Cronología de MySQL - Historia de MySQL | Solo Codigo Web*. [online] Available at: http://www.solocodigoweb.com/blog/2014/03/04/una-breve-cronologia-de-mysql/ [Accessed 24 Sep. 2016].

Telegram. (2016). *Telegram F.A.Q.*. [online] Available at: https://telegram.org/faq [Accessed 8 Feb. 2016].

Xatakandroid.com. (2015). *¿Cómo compartir nuestra localización en cada momento con Android?*. [online] Available at: http://www.xatakandroid.com/tutoriales/como-compartir-nuestra-localizacion-en-cada-momento-con-android [Accessed 17 Jul. 2016].

10. Apéndice:

10.1. Acrónimos:

- **LNR**: La nevera roja.
- **JE**: Just Eat.
- **SO**: Sistema operativo.
- **AES**: Advanced Encryption Standard.
- **SMS**: Short Message Service.
- **SQL**: Structured Query Lenguage.
- **GPS:** Global Positioning System.
- **API:** Application Programming Interface.
- **TPV:** Terminal punto de venta.

10.2. Definiciones

- **Script**: Programa simple normalmente, utilizados generalmente para interactuar con el sistema operativo o el usuario.
- **LUA**: Lenguaje de programación ligero, se ha creado con la idea de que sea un lenguaje para la implementación de Scripts. Es software libre.
- **Man-in-the-middle:** Tipo de ataque al proceso de comunicación entre dos agentes que consiste en interceptar los mensajes que estos se envían con la intención de hacer creer a ambos que el atacante es el otro interlocutor.
- **Bot**: Es un programa que se encarga de realizar acciones imitando a un humano. Estas acciones pueden ser repetir una acción continuamente o mantener conversaciones.
- **Python**: Lenguaje de programación centrado en la legibilidad del código.
- **Parsear:** Consiste en el proceso de transformar información de tal forma que podamos utilizarlo como entrada de una función de un programa.
- **Root:** Es el nombre que recibe el usuario de un equipo basado en un sistema operativo UNIX que tiene todos los derechos en cualquier modo.

10.3. Conclusión, introducción en ingles:

10.3.1. Introduction:

Consumers are increasingly using their cell phones for many aspects of their daily life. As a result, companies are developing more applications than can be accessed by their SmartPhones no matter where they are.

- A large part of consumer use of these cell phones is realted to instant messaging.
- Recently, web services used to order food for home delivery have been very successful.

The system that is being presented takes into account this growing demand, and establishes in a new way for the consumer to contact with the company/restaurant, by ordering food for home delivery using the instant message application *Telegram*.

The system permits clients to rapidly order, using a minimum of words. Consumers can also take the time they need to discuss which dishes to order, without the pressure of having the person taking the order at the restaurant waiting on the phone for their decisions.

The managers of food establishments will have a more efficient way of taking orders, with less need for personnel, thus becoming more competitive, and permitting a more direct management of orders with a choice of applications.

The system is to be installed in restaurants, and works with few technical requirements. Thus, future problems created by technological progress or technical difficulties can be avoided. (The system works in digital devices costing approximately 40 euros). In fact, future technological advances could only increased the ease and speed of the application.

The system is designed for the use of fast food establishments, whether or not they offer home delivery. Since attracting the clients of these establishments is essential for the application to be useful, it is important to comment on their clientele.

The clientele is young, (adolescents and young adults) and thereby very familiar with the use of applications on Smart Phones. Given that a lack of familiarity with the use of instant messaging is the main obstacle to the use of our system by the clientele of fast food establishments, It is important that the use of our system be as easy and comfortable as the use of WhatsApp or other instant messaging systems.

Although other age groups can be clients of fast food restaurants, our application will initially be directed towards the groups that already feel comfortable with the use of Smartphones for various activities, including holding conversations via screen, as opposed to speaking directly.

We can divide the potential users of our application into two groups.

1. The first group is made up of family units, in which at least one member will be the initial client, who will be in charge of placing the order for the other members of the group. Our system must be accessible enough to attract other familiy members, permitting its use by another family member the next time an order is placed. Thus “mouth-to-mouth” information transmitted among familiy members will increase the system’s acceptance.

2. The second group is made up of groups of people that are not family units. These can range from friends, to workplace companions. In general these groups will exhibit a higher degree of homogeneity in age and as regards familiarity with information technology.

The personal data of those clients that place the order will be protected by current legislation, specifically the la Ley Orgánica 15/1999 of December 13th.

10.3.2. Objetives:

The main goal of the project is the elaboration of a *SCRIPT* for *Telegram* that can be implemented in real establishments, permitting the generation of profit.

Given our lack of experience in this field, learning how to use the needed instruments forms part of the objectives, together with learning best practice principles of programming and development. The community of developers of LUA will be consulted, as will be the availabe bibliography.

The knowledge acquired will be used to develop a system capable of interating with an operative system, an external program and a data base.

10.3.2.1. Specific goals :

3. To elaborate a state of the art that explains the chosen design.

4. To describe the requirements that the system design must meet

5. To develop the project’s planning and budget

6. To describe possible future derivations of the Project

7. To explain the design based on the user interphase, and the arquitecture and security of the system.

8. To situate the Project in the current socioeconomic, legal and technological context.

10.3.2.2. Material and Methods:

A computer and a cellphone have been necessary, and the app *Telegram* installed. The phone used Android, although the app works with different cellphone operating systems.

10.3.2.3. The Computer:

Operating Systen	9.	Ubuntu Linux 16.04
CPU	10.	Intel Core i7-6700HQ
RAM	11.	8.00 GB Dual-Channel
Graphics	12. 13.	Nvidia GeForce GTX 960M 2GBs Intel HD Graphics 530
Storage	14.	1TB HDD

Tabla 84: Computer specs

10.3.2.3.1. Cell phone:

Various cell phones have been tried out to assure the system functions when various users are interacting with it.

Any cell phone that permits the use of the app *Telegram* can be used for this interaction. Based on the Operating System of the phone, different versions can be used.

- **Android**: Version 2.2 and superior.
- **iOS**: 6 and later.
- **Windows Phone**: No specifications are given regarding which versions cannot use the app correctly.

10.3.3. Conclusion:

The system that has been implemented permits the management of orders for restaurant home delivery. It meets all the requirements and needs initially stipulated when initiating its design. It permits a natural and easy interaction with the client, and offers restaurant employees a user-friendly interphase and quick access to required information.

The system is operative, and ready to install in any interested establishment. Prior to installment, the restaurant owner can decide which specific adjustments he is interested in.

The experience of using a new language, and a code from a repository with sparse documentation has been arduous, yet enriching.

The experience has permitted learning in what consists a development Project, first hand and from an integral point of view, thus applying many of the capacities separately learned during the Degree, and integrating them, interacting with a data base from an external program, assuring an automatic interaction by using one of the operating systems studied.

A system has been developed that can be converted into a commercial product, similar to applications that have been developed to facilitate the interaction between restaurants and their clients. Therefore, the system could be completed to become a serious competitor in the food home delivery market.

During the development process, several potential modifications and new lines of development have been identified. Several of them are included in the section “Future improvements.”

The possibility of interacting with a Linux-based system, as the user orders using his device, opens many new possibilities, from captures by camera that can be sent by cell phone, changes in the computer simply sending a message from the cell phone, up to creating a blog where users can interact among themselves and with the system Via *Telegram.*

In conclusión, through the development of this system I have become familiar with useful tools that permit the development of projects that currently do not exist on the market. The developed system could have commercial applications in the future given the system’s novelty, and its close relationship withsocial networks, a market that is still on the rise.

10.3.4. Examples of use:

In contast with the interphase that is presented to the restaurant clients, the messages that the restaurant employees will use are designed in such a way that they can be written very quickly, although they are not written in a natural language. For example, to consult the list of orders that have yet to be filled, their dinetifier and the time at which the order was placed, and “h” alone will be written.

When the system receives a message that is not expected, the system responds as follows:

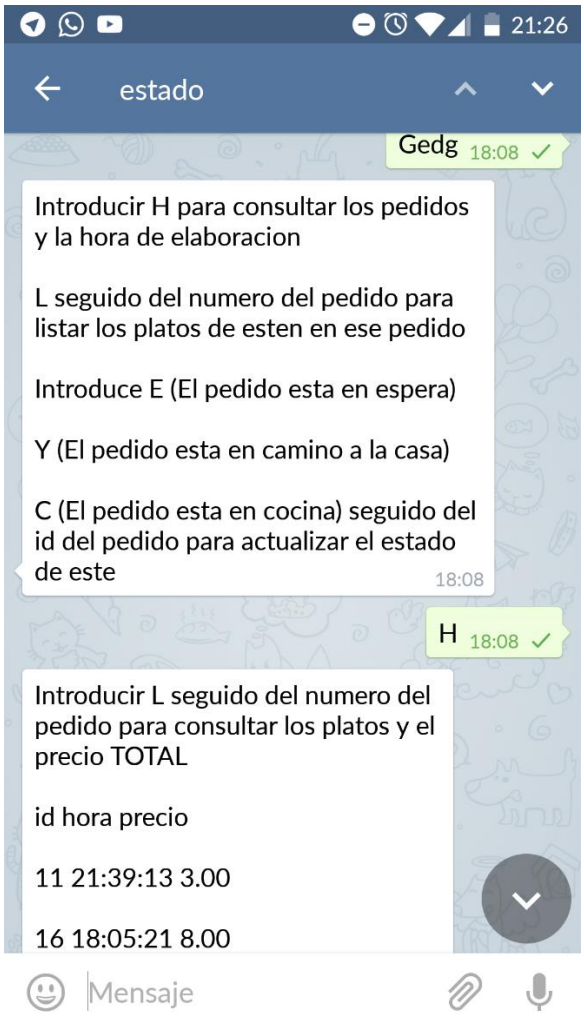


Figura 47: Avaliable commands for gerente screen

To access more detailed information on an order, the identifier given by the system when “h” is written is necessary.

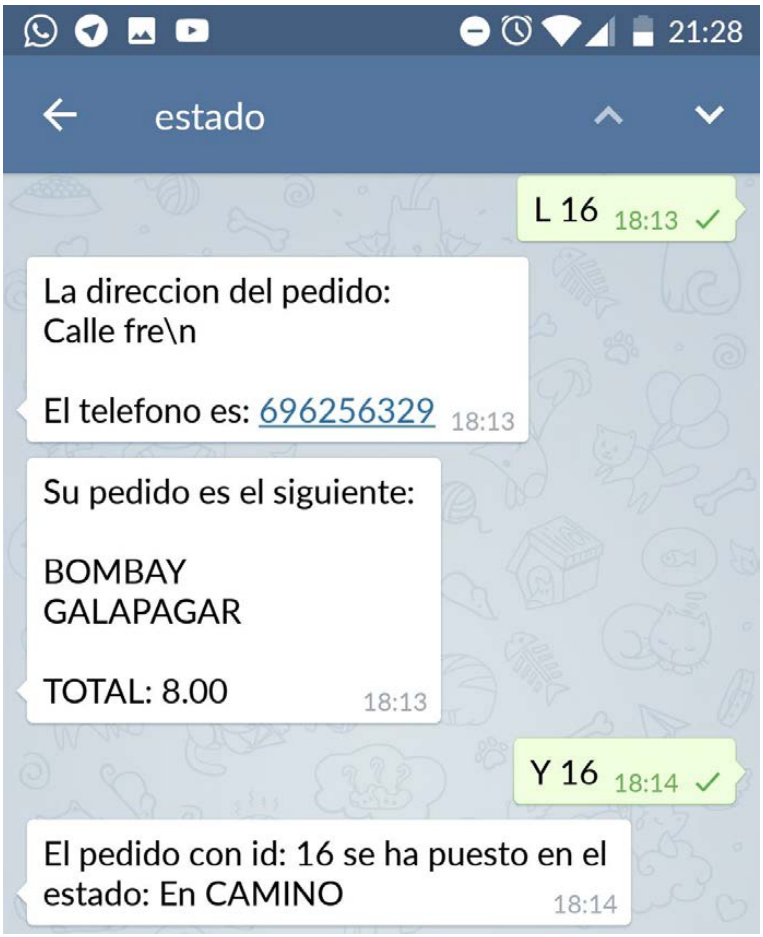


Figura 48: Order details screen

This identifier permits changes in the state of the order as its preparation advances, or indication that it has been delivered.

Costumers are users of the system that place orders through it to any establishment that has installed the system.

The system keeps information on clients' orders simply for organization purposes. The correct use and custody of the data is the responsibility of the manager of the establishment.

The first message that will be received if no preestablished command is entered, will be the presentation of the establishment, and informtion on what messages can be introduced, and what responses the user/costumer can receive.

Initially, the system will ask the costumer whether the order will be home delivered or whether the customer will pick the order up at the restaurant. If the user choses "home" the system waits for the customer to introduce the address the food will be delivered to, and confirm it.

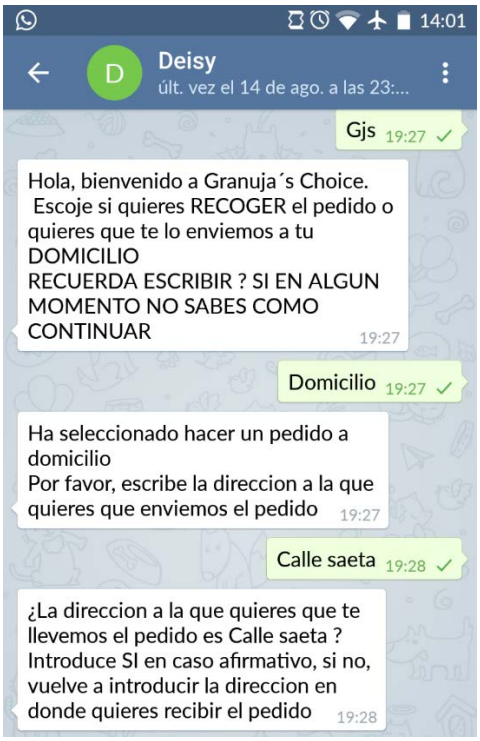


Figura 49: Start screenn of a home delivery

Once the costumer has confirmed that the entered address is correct, the process of ordering starts. The image below illustrates the confirmation of the address and the selection of the dishes chosen from the menu when the customer is familiar with the menu.

Figura 50: Adding dishes screen

When the costumer introduces a message that does not coincide with what is in the data base, the system will tell him and write once more what messages can be introduced and what each message will produce.

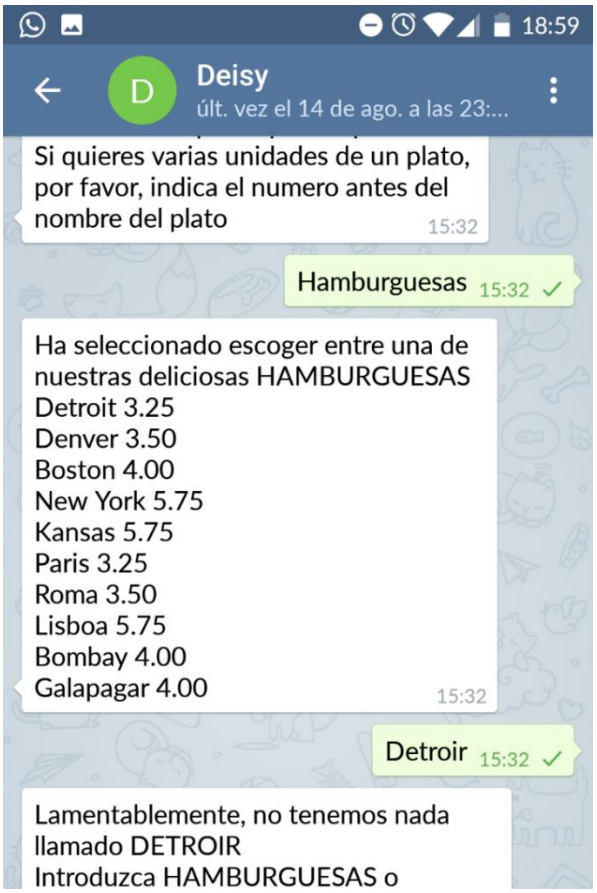


Figura 51: Listing dishes example

If a customer wishes to eliminate a dish from the order, he will only have to write “eliminar” followed by the name of the dish to be dropped from the order. The system will answer confirming that the dish has been dropped.

Figura 52: Deleting process example

No additional confirmation message for eliminating an order will appear, since it is difficult to introduce “eliminar” followed by a specific dish by mistake.

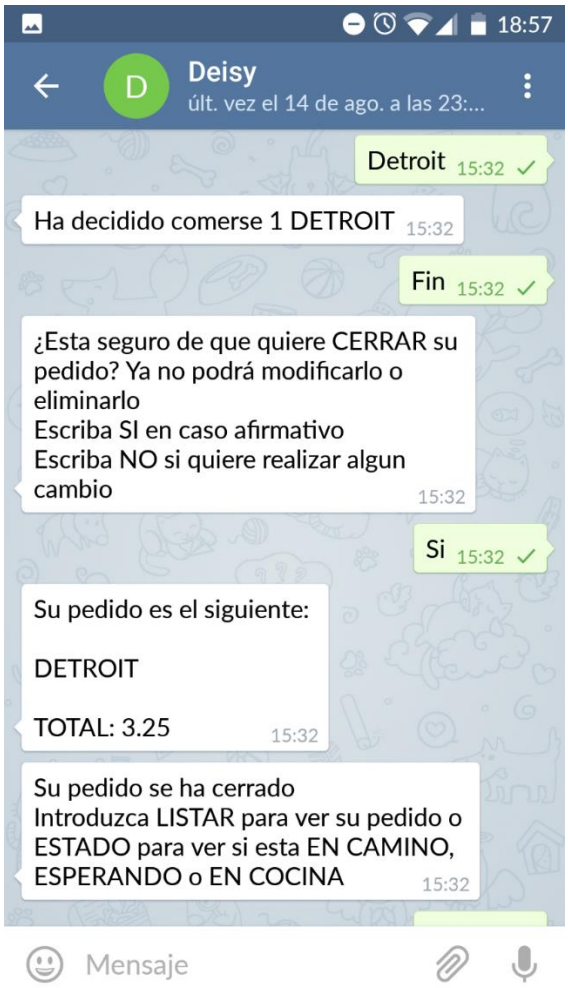


Figura 53: End of the order screen

Once the customer has added on all of the dishes he wants to order, he will simply have to add on “end” and confirm that he wishes to place the order as it is.

He will be asked to confirm that the order is complete. Once confirmed, the order appears on the interphase of the restaurant’s cooks so that they may start is preparation

Once the order has been placed, the system awaits information on the state of the order.

The phone number used to place the order will not be able to initiate a new order until the initial order has been delivered, since the order is not closed until it has been marked as delivered. It then disappears from the current order list and is copied onto the “finished” list.

10.3.5. Future areas of development:

10.3.5.1. Sending automatic updates on the state/situation of the orde :

This improvement consists in adding code to the script code that informs the person that places the order of changes to the state of the order, such as " being cooked," or " already finished" or " on its way" automatically.

To send these messages automatically, the name of the person that places the order would be required, and thus would have to be stored together with the phone number. Thus, another attribute would have to be included in the customer's table.

Once the table has been created and the user's name has been recorded, the system would access this information and could write to the customer automatically as the order undergoes changes.

This modification could be extended to inform on the state of each of the dishes ordered, with a final message when the entire order is completed and ready to be delivered. For this, a new attribute would be introduced into the Table "Cuenta-Producto" to represent in which state the product currently is in.

10.3.5.2. Text recognition "did you mean:

To permit constant feedback from the system to the customer, and make sure that system use is in consonance with best pratices in users' interphase when receiving a message that is not recognized, the system will reply with messages that will change the state and give information regarding every single change generated.

This type of interaction when correctly applied assures that the experience of the costumer when going throught the whole process is positive. However, entry errors, i.e. incorrect spelling of the name of a dish, or sending the wrong command, could cause problems.

These problems could be corrected by adding a functionality which "guesses" what the customer meant. Similar to what ocurrs when using the search engine "Google," we could make the interaction more dynamic if the system answers with suggestions, thus avoiding the need for the customer to rewrite the entry.